UAH

Diseño, implementación y evaluación de un sistema de diálogo persona-máquina en un espacio inteligente

Máster Universitario en Sistemas Electrónicos Avanzados
Sistemas Inteligentes
Departamento de Electrónica

Presentado por:
David Casillas Pérez

Dirigido por:
Marta Marrón Romera y Javier Macías Guarasa

Alcalá de Henares, a 24/25 de septiembre de 2014

A todos aquellos que tomaron por máxima la buena voluntad.

...

"increscunt animi virescit, volnere virtus"
Friedrich Nietzsche

Agradecimientos

Ayuda a tus semejantes a levantar su carga, pero no te consideres obligado a llevársela.

Pitágoras

La elaboración de este proyecto no ha sido una tarea fácil. Conciliar la vida laboral que comencé a principios de año junto con la vida académica y la vida familiar es un juego de malabares en el que poco a poco se van incrementando el número de pelotas y en el que sabes que algún día se caerán. Muchas son las personas que han hecho posible que este documento vea hoy la luz y aunque su rúbrica no aparezca en este documento deben ser mencionadas:

En primer lugar mis padres. Es imposible aguantar tanta presión, evitar tantas puntas de lanza sin su apoyo. Ellos son y han sido desde siempre la razón por la que hoy esté escribiendo estas palabras de agradecimiento.

A mi hermano y a mi tía le debo agradecer su esfuerzo que hacen por enseñarme el mundo que hay más allá de la física, la ingeniería y las matemáticas. Ellos son quienes me acercan la historia, el arte y la música, quienes me humanizan.

A mi novia debo de agradecer su capacidad de aguante, su tolerancia y comprensión. Ellas siempre está ahí para darme se apoyo en los momentos dificiles y me consigue arrancar una sonrisa. Por otro lado, disfruta viendo como poco a poco logro cada hito que me propongo y eso hace que me sienta bien.

A mis tutores porque son los que me han llevado de la mano en esta ultima etápa de mi formación, de ellos he aprendido muchos conocimientos y lo que es más importante, el razonamiento crítico para evaluar todo aquello que construyo como ingeniero. El proyecto que hoy escribo es en parte suyo y por ello les doy las gracias.

No me olvido del resto de mis amigos y compañeros que me han ayudado, pero es imposible hacer mención a todos ellos. De todos se aprende algo y eso es lo más importante.

Resumen

El trabajo descrito en este documento consiste en el diseño, la implementación y la evaluación del sistema de diálogo inserto en un interfaz de control multimodal persona-máquina que controla un espacio inteligente. El sistema de diálogo diseñado en este proyecto es una evolución considerable del módulo de comprensión desarrollado en los proyectos fin de carrera de Manuel Villaverde Díez y de David Casillas Pérez titulados *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control vocal* [1] y *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control gestual* [2]; proyectos conjuntos que pretendían desarrollar un interfaz de control de un espacio inteligente por medio de comandos orales y gestuales.

Los sistemas de diálogo persona-máquina son, a día de hoy, un problema ni mucho menos resuelto. La dificultad fundamental en el diseño e implementación de este tipo de sistemas radica en la necesidad de que muestren un comportamiento "natural", haciendo frente a la fuerte variabilidad de las interacciones que una persona produce, y también a los posibles errores de los sistemas de reconocimiento automático de habla de los que obtienen las frases que más probablemente ha generado el usuario.

A lo largo del proyecto se describe la construcción de un sistema de diálogo siguiendo una serie de pasos entre los que se destaca la elección del vocabulario y de la gramática de generación de sentencias en función de la aplicación, la definición de los principales errores a detectar y corregir, el desarrollo del algoritmo de ejecución que corregirá los errores definidos y solicitará al usuario la información necesaria en cada caso, la implementación de una técnica específica de conformación por parte del usuario como es la cancelación y el diseño del módulo de generación de respuestas parametrizado, el cual se comunicará con el usuario.

Palabras clave: Sistema de diálogo, modulo de comprensión, interfaz hombre-máquina, espacio inteligente y audiovisual.

Índice general

Índice de figuras

Lista de acrónimos

GUI	Graphical User Interface.
HCI	Human-Computer Interface.
HMM	Hidden Markov Model.
LIFO	LAST IN FIRST OUT.
NUI	Natural User Interfaces.

Capítulo 1

Introducción

El mayor problema en la comunicación es la ilusión de que se ha logrado.

George Bernard Shaw

1.1 Presentación

La comunicación es un concepto intrínseco en los seres humanos, hasta tal punto que forma parte de nuestra esencia y de nuestro ser. La comunicación está presente en todas las etapas de nuestra formación, durante la infancia, la adolescencia, la madurez y también en las últimas etapas de nuestra vida. La comunicación es la causa de que se formaran sociedades, naciones, de los grandes descubrimientos de toda nuestra civilización y pese a ser tan importante, tan fundamental, apenas somos conscientes de los procesos fisiológicos que se producen en nuestro cerebro cuando nos comunicamos, de los conceptos que se forman a partir de sonidos, imágenes, gestos que percibimos desde fuera; del aprendizaje que a partir de estos conceptos se produce. Solamente por medio del estudio de los procesos que llevan a establecer una comunicación entre personas podemos construir sistemas que traten de imitar un proceso tan complicado como es la comunicación entre personas.

Desde que nos levantamos hasta que nos acostamos las personas se comunican entre ellas con el objeto de intercambiar información. A todas horas, con tus padres, con tus amigos, con gente con la que no compartes un vínculo afectivo, con todos ellos compartimos información útil para el desarrollo de nuestra vida. Pero, hasta las conversaciones más fútiles, las más insignificantes se hacen complejas a la hora de ser trasladadas al ámbito de las máquinas. Realizar un sistema de diálogo general que comprenda todas las conversaciones en todos contextos posibles tal y como los humanos hacemos a diario está lejos de ser logrado. Tan solo, el establecimiento de determinadas condiciones y límites de uso permite la construcción de sistemas con diálogos relativamente complicados.

Los sistemas de diálogo tienen como objetivo facilitar la interacción mediante el denominado lenguaje natural entre las personas y las máquinas. En este sentido, los sistemas de diálogo se enmarcan dentro del ámbito de la comunicación entre personas y computadores *Human-Computer Interface (HCI)*, concretamente en el ámbito de las *Natural User Interfacess (NUIs)*. El habla y los gestos son candidatos a ser utilizados como complemento o sustituto de la clásica interacción que se venía utilizando para el manejo de las máquinas, y su estudio sigue ocupando una parte importante de la actividad científica del momento. En el futuro, se prevé que la interfaz hombre-máquina se transforme en una interfaz que permita la libre movilidad del individuo, desligándose cada vez más de periféricos que requieran para su

funcionamiento de un contacto físico. En este estudio de interfaces de tipo NUI los sistemas de diálogo tienen especial importancia.

En el ámbito de los espacios inteligentes, el estudio de los sistemas de diálogo como pieza clave para la comprensión de las sentencias (ya sean orales, gestuales,...) es un tema hoy en auge. Es este contexto, surgen problemas como los acarreados por los sistemas de reconocimiento (los cuales normalmente toman como entrada), los problemas que inherentemente contiene el habla espontánea: elipsis, anáforas, problemas deícticos..., las necesidades de estrategias de verificación. La definición de un conjunto de instrucciones robusto y sin ambigüedad, con el cual nos dirigiremos al sistema de diálogo, es también un paso importante tanto para el reconocimiento como para el diseño de los sistemas de diálogo.

Actualmente, dentro del grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte del Departamento de Electrónica de la Universidad de Alcalá, se viene trabajando en una línea de investigación cuyo objetivo es permitir un análisis tanto de la información gestual como de la información audible que puede generar un sujeto, y a partir de la misma, obtener información que pueda resultar útil para controlar un entorno inteligente. En este punto el sistema de diálogo puede verse con dos perspectivas diferentes:

1. El sistema de diálogo es un sistema que incluye a los reconocedores y los sintetizadores como parte del sistema. Con esta perspectiva el sistema de diálogo se entiende de forma global como sistema que es capaz de recoger la información emitida por el usuario en cualquiera de sus formas(oral, gestual,...), entenderla, y emitir una respuesta que pueda ser comprendida por el usuario(normalmente la respuesta se genera de la misma naturaleza que la información de entrada: oral, gestual, textual...).

2. El sistema de diálogo no incluye a los reconocedores dentro del propio sistema a diferencia del bloque funcional anterior. Tampoco incluye a los sintetizadores como parte del mismo, sino que son entendidos como un bloque funcional de salida. Así entendido, el sistema de diálogo toma como entrada las sentencias que el reconocedor a transcrito, y proporciona a la salida tanto las sentencias comprendidas que pueden ser pronunciadas por un sintetizador, como información que será ejecutada por otros módulos, como por ejemplo el de virtualización.

El proyecto fin de máster que se presenta surge como una ampliación del módulo de comprensión, una de las piezas clave del interfaz de control multimodal desarrollado en los proyectos fin de carrera de Manuel Villaverde Díez y de David Casillas Pérez titulados: *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control vocal* [1] y *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control gestual* [2]. Aunque se escribieron dos proyectos, el trabajo que se propuso realizar formaba parte de una iniciativa coordinada de los dos proyectos anteriores orientados al diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente, que permitía a un usuario controlar una serie de elementos móviles y fijos situados en un espacio virtual que modelaba un espacio inteligente real. Estos proyectos fin de carrera abordaban muchos aspectos relacionados con los espacios inteligentes, algunos muy diferentes. Por este motivo, el interfaz de control no disponía de un sistema de diálogo como tal si no más bien, un módulo comprensor que recibía las sentencias por parte de los reconocedores y decidía si eran frases ejecutables o no por el sistema.

Este proyecto fin de máster es por tanto heredero de ese módulo, y se centrará en crear un sistema de diálogo real. El proyecto fin de máster, en concreto, se centrará en las labores relacionadas con el diseño, la implementación y la evaluación del sistema de diálogo sistema de diálogo, y de aquí su titulo: *Diseño, implementación e interpretación de un sistema de diálogo persona-máquina en un espacio inteligente.*

El proyecto tratará las tareas de diseño del vocabulario y de la gramática necesarias para la aplicación y seguirá con la definición de los errores más comunes que los usuarios pueden cometer y la resolución del problema de la detección de dichos errores así como su corrección. Además, planteará una serie de técnicas que ayudarán al sistema de diálogo a favorecer la comunicación con los usuarios que lo utilicen, como lo son las técnicas de confirmación o las de generación de respuestas parametrizada en base a roles combinándolas con un aleatorizador.

1.2 Motivación

Entre las motivaciones más importantes que nos han impulsado en el desarrollo de este proyecto, debemos citar las siguientes:

- Desarrollo de una de las piezas clave para el desarrollo de interfaces de control que permitan establecer comunicación entre máquinas y personas discapacitadas.
- Trabajar en el desarrollo de interfaces intuitivos y basados en el lenguaje natural de las personas.
- Aprender, mientras resolvemos un problema real, multitud de herramientas.
- Enfrentarnos con madurez a los problemas surgido durante la implementación del proyecto.
- Trabajar en la integración de sistemas que pueden ser de muy distinta naturaleza.
- Adquirir nuevas habilidades relacionadas con la investigación y la búsqueda de soluciones.
- Y, por que mentir, alcanzar finalmente el título de máster.

1.3 Contextualización y breve descripción del trabajo

Como ya hemos dicho en el apartado anterior, este trabajo trata de mejorar uno de los módulos que se desarrolló en los proyectos titulados: *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control vocal* [1] y *Diseño, implementación y evaluación de un interfaz de control multimodal en un espacio inteligente: control gestual* [2]. Véase la figura 1.1.

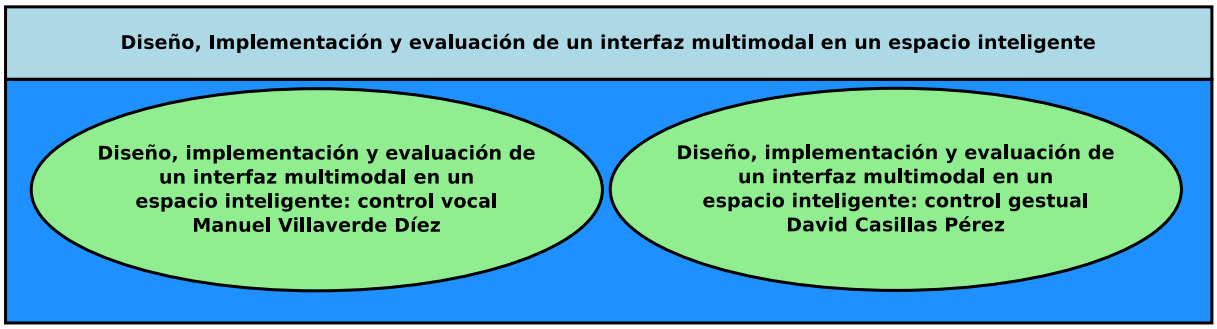


Figura 1.1: Descomposición del trabajo final: el interfaz de control multimodal.

Nos referimos al módulo de comprensión, un módulo del interfaz que se encargaba de extraer del mensaje emitido por el usuario la información necesaria para que los actuadores del interfaz lleven a cabo las tareas requeridas. La imagen de la figura 1.2 muestra un esquema funcional de este módulo.

Al tratarse de un proyecto que parte de un trabajo previo es conveniente hacer un resumen de dicho trabajo para contextualizarlo. Por ello, la siguiente sección explica qué es un interfaz de control multimodal y cómo se enmarca el sistema de diálogo dentro del mismo:

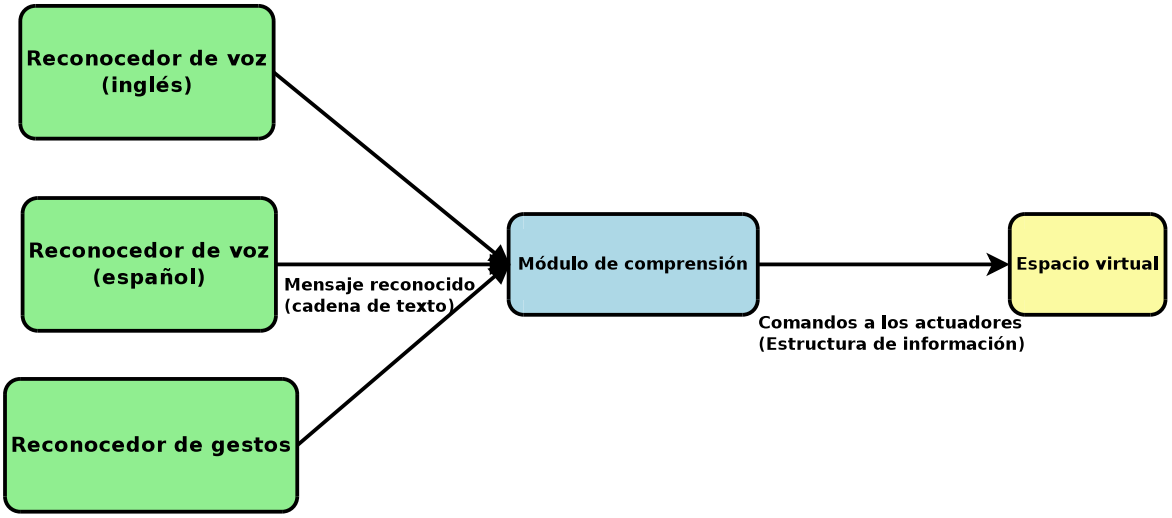


Figura 1.2: Esquema de trabajo del módulo de comprensión.

1.3.1 El interfaz de control multimodal

En los proyectos citados previamente, el objetivo fundamental perseguido por sus autores fue el desarrollo de un interfaz de control multimodal que permitiese a los usuarios controlar un espacio virtual mediante comandos de habla y gestos. Los proyectos tenían como meta conseguir una comunicación lo más natural posible entre una persona y un entorno inteligente. Un interfaz de comunicación hombre-máquina es un sistema que permite el intercambio de comunicación entre el hombre y la máquina mediante hardware y software. El objetivo fundamental de este tipo de interfaces es hacer que este intercambio sea lo más eficiente posible: minimizar errores, incrementar la satisfacción del usuario, disminuir su frustración, y hacer más efectivas las tareas que rodean a las personas y a los computadores [3]. Como ejemplos típicos, podemos citar los más extendidos como los teclados, los ratones, los monitores,... Cuando hablamos de estos dispositivos nos referimos no sólo al dispositivo físico, sino también, al conjunto de librerías y aplicaciones que hacen posible su comunicación con la máquina.

Los interfaces hombre-máquina han evolucionado de forma considerable a lo largo de la historia. Los primeros interfaces trataban de establecer la comunicación hombre-máquina a través del contacto físico con un dispositivo, sobre todo en interfaces de entrada de datos. Ejemplos de ellos son el teclado y el ratón. Hoy este tipo de interfaces siguen siendo los más extendidos para el control de las máquinas (véase la figura 1.3), aunque la tecnología posibilita otro tipo de interfaces basados en el lenguaje natural.

(a) Ratón [4] (b) Teclado [5]

Figura 1.3: Interfaces clásicas de comunicación hombre-máquina.

Los interfaces naturales de usuario o NUI son interfaces que interactúan con el sistema, aplicación,...,

sin utilizar sistemas de mando o dispositivos de entrada de las interfaces gráficas de usuario o *Graphical User Interface (GUI)* como sería un ratón, un teclado alfanumérico, un lápiz óptico, un *touchpad*, un *joystick*,... y en su lugar, se hace uso de movimientos gestuales con las manos, o con el propio cuerpo; o se realiza un control por medio del habla humana.

La clave de estas interfaces es la ausencia de contacto físico con los dispositivos para establecer comunicación entre el hombre y la máquina, y el uso del lenguaje natural por medio de habla y gestos. En la imagen de la figura 1.4, se muestra un interfaz de comunicación inteligente basado en los principios del lenguaje natural: concretamente utiliza un control oral y gestual.

Figura 1.4: Interfaz de comunicación inteligente [6].

Centrándonos en nuestros proyectos, se diseñó un interfaz de control vocal y gestual que actuaba sobre un espacio inteligente muy concreto. El espacio inteligente diseñado estaba alojado en la sala *geintra-ispace* donde había un conjunto de elementos (una puerta, una televisión, unas ventanas, etc. mostrados en la figura 1.5) sobre los que se podía ejercer alguna acción, y una serie de sensores con los que se capturaba la información del usuario en escena. Mostrados en la figura 1.6

El interfaz de control multimodal trata por tanto de recibir información por parte de los sensores, procesarla, y elaborar las respuestas que ejecutarán los actuadores del espacio inteligente. El espacio inteligente diseñado tiene el diagrama de bloques funcional de la figura 1.7. Explicaremos brevemente a continuación, cada uno de los bloques funcionales del diagrama de tal modo que podamos hacernos una idea del funcionamiento del interfaz y, por tanto, del contexto donde el sistema de diálogo estará inmerso.

- Como se observa en la figura 1.7, los primeros bloques funcionales del diagrama se corresponden con los sensores. Los sensores son los elementos del interfaz de control que permiten capturar el

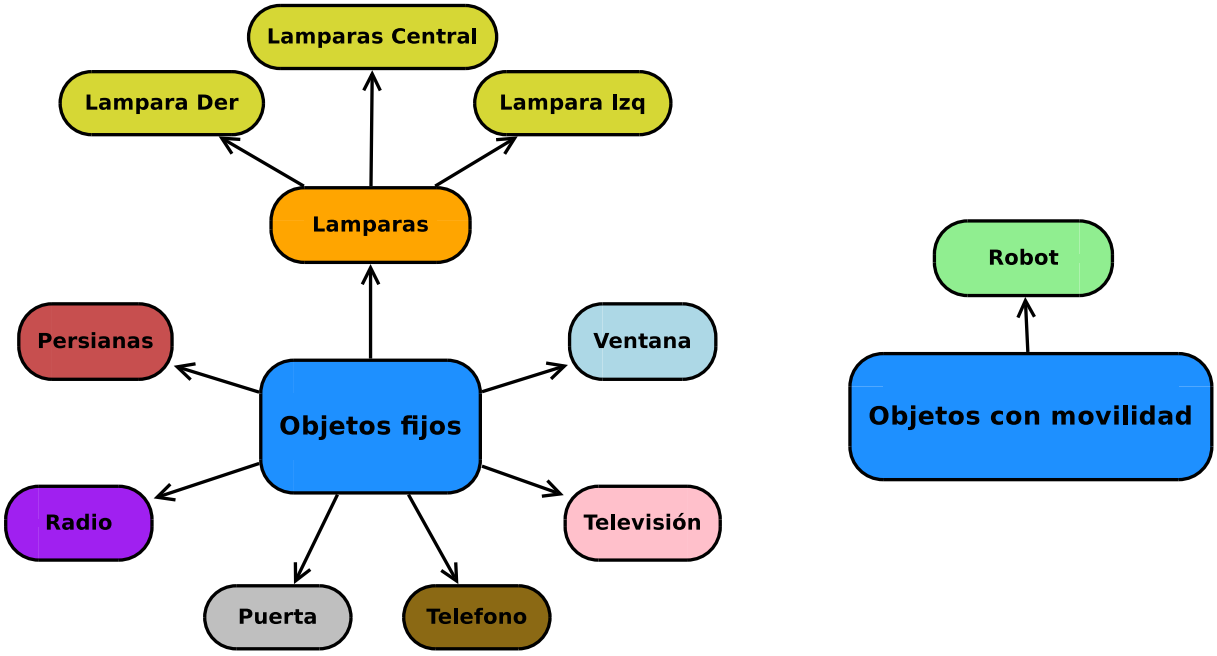


Figura 1.5: Actuadores del espacio inteligente que controlaremos.

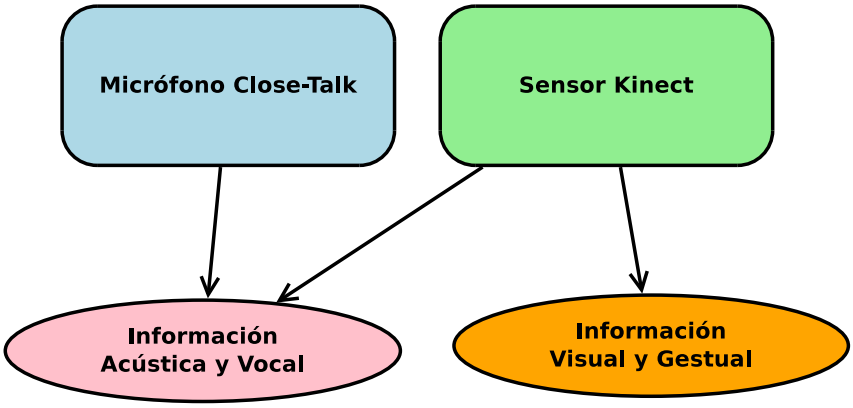


Figura 1.6: Sensores del espacio inteligente que controlaremos.

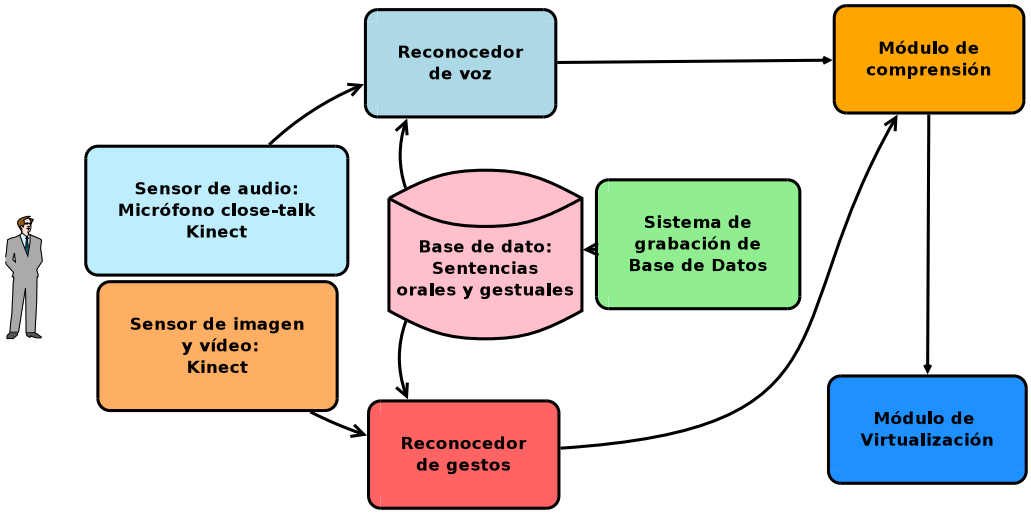
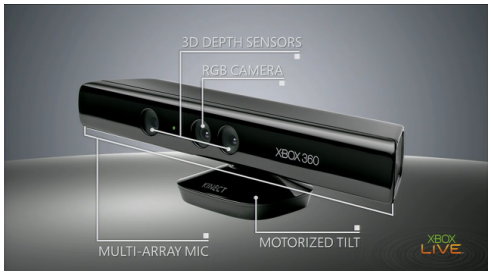


Figura 1.7: Diagrama de bloques del interfaz de control multimodal.

sonido o los gestos que portan el mensaje que deberá ejecutar el interfaz. En los proyectos de David y Manuel, los sensores utilizados fueron el sensor Kinect (véase la figura 1.8a) y el micrófono *closs-talk* (véase la figura 1.8b). El sensor Kinect recoge la información gestual por medio de su sensor de profundidad y proporciona al reconocedor información para la clasificación de los gestos. Del mismo modo, el micrófono *close-talk* realiza el mismo proceso pero con las sentencias audibles.



(a) Sensor Kinect [7].

(b) Imagen del micrófono *close-talk Logitech Wireless Headset H760* [8].

Figura 1.8: Sensores utilizados en el interfaz de control inteligente.

- Los reconocedores de voz y de gestos obtienen a la salida la sentencia transcrita del comando de voz o la sentencia gestual que el usuario ha pronunciado o ha gesticulado y cuya información ha sido recogida por los sensores. Tanto el reconocedor de voz como el de gestos ha sido diseñado utilizando los *Hidden Markov Models (HMMs)*. Para más información acerca de la teoría de los HMMs consulte los capítulos teóricos de los proyectos de Manuel[1] y de David[2]. Si desea profundizar en el tema, autores como Rabiner[9], Lee[10] o Baum[11] deben ser leídos. La construcción de un reconocedor de gestos o de voz se sale de los objetivos de este documento, no obstante en la figura 1.9 se muestran los principales bloques en los que se divide.

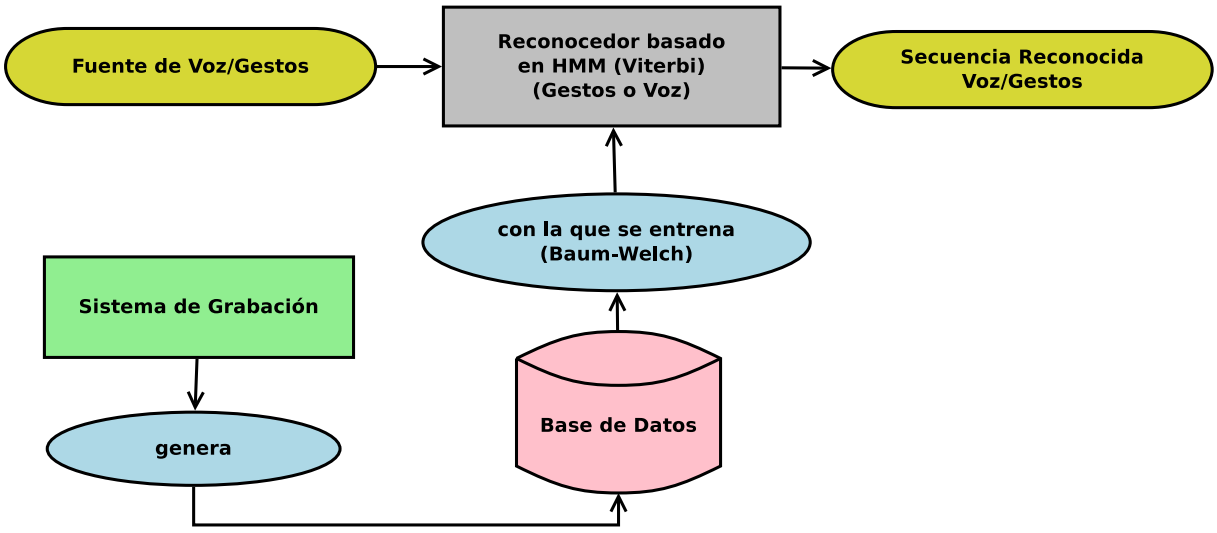


Figura 1.9: Esquema funcional del reconocedor y del sistema generador de la base de datos.

- La base de datos que se visualiza en la figura 1.7 es fundamental durante el entrenamiento de los HMM que constituye el reconocedor. Esta base de datos está formada por sentencias gestuales y orales comunes que se pueden formular en el espacio inteligente. La base de datos generada contiene la suficiente información como para entrenar todos los modelos HMM que conforman el vocabulario y la gramática de los reconocedores. Es importante que la base de datos contenga un buen número de ejemplos para un correcto funcionamiento por parte de los reconocedores.

- Para construir la base de datos, se desarrolló un sistema generador de base de datos que permite grabar bases con sentencias tanto oral como escrita de forma tanto secuencial como simultánea. Esta aplicación fue desarrollada en los trabajos de David[2] y de Manuel[1].

- El módulo de virtualización es un interfaz gráfico desarrollado con la librería *OpenGL* con el objetivo de representar el propio interfaz inteligente de forma real. El módulo representa la sala *geintra-ispace* (véase la figura 1.10, que es la sala física donde se aloja el espacio inteligente) en donde aparecen alojados los actuadores (véase la figura 1.5) y los sensores (véase la figura 1.6). Se utiliza para comprobar la ejecución de las sentencias que emite el usuario en el espacio inteligente, debido a que no contamos con un espacio inteligente real en donde ejecutar el interfaz. La imagen de la figura 1.11 muestra esta módulo.

Figura 1.10: Sala *geintra-ispace*.

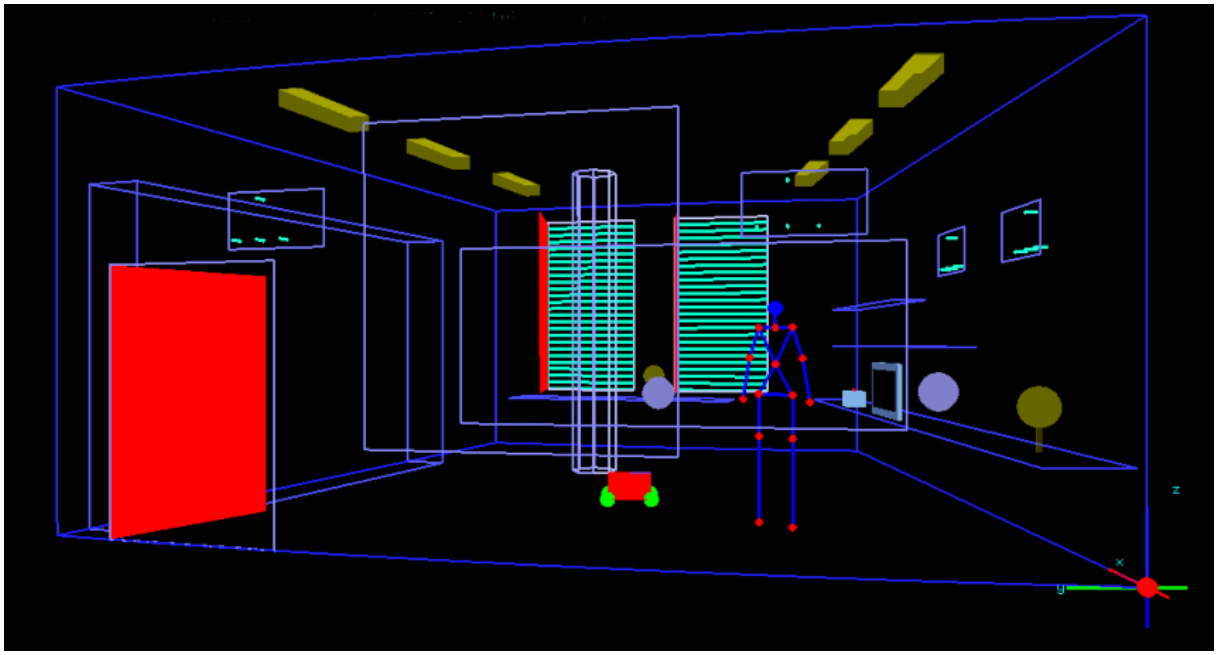


Figura 1.11: Espacio virtual que simula el espacio inteligente real.

Hasta este punto no hemos hablado del módulo de comprensión, módulo del que el sistema de diálogo supone una evolución. En la siguiente sección 1.3.2 se resumirán las tareas que este módulo desarrollaba

en los proyectos de Manuel[1] y David[2] para que nos hagamos una idea del punto desde el que partirá el sistema de diálogo diseñado en este proyecto fin de máster.

1.3.2 El módulo de comprensión

El módulo de comprensión es el módulo que elabora las órdenes pertinentes que se deben ejecutar sobre el módulo de virtualización del espacio inteligente una vez se haya comprendido la sentencia. Si hubiésemos implementado un interfaz de control con sólo los reconocedores de voz y de gestos, al sistema le faltaría la capacidad comprensora que permite al interfaz entender verdaderamente lo que el usuario ha querido decir, y ejecutar el mensaje recibido en consecuencia. El ser humano, al recibir información a través de los sentidos, elabora respuestas a partir tanto del significado reconocido, como del contexto en el que se produzca la recepción de la información, es decir, en función de la situación que rodea la percepción de dicha información. En un escenario general dentro del espacio inteligente, un usuario podría decir sentencias desordenas, erróneas, o simplemente que se alejan de la concepción de sentencia ideal que las reglas de la gramática definen. El módulo de comprensión, teniendo presente la gramática, detectará los posibles errores sintácticos, las elipsis temporales abundantes en el habla, las repeticiones...

Resumiendo, el módulo de comprensión realiza labores de contextualización y de análisis a partir de las secuencias de palabras reconocidas, comprende el mensaje y elabora las respuestas que hay que ejecutar. En la figura 1.12 se muestra un esquema completo de la funcionalidad de este módulo. Para más información, consulte el capítulo dedicado a este módulo en los trabajos fin de carrera de Manuel[1] y de David[2].

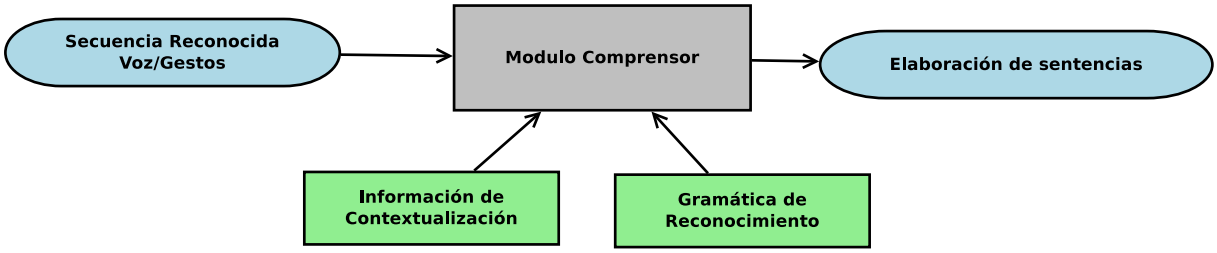


Figura 1.12: Esquema funcional del módulo comprensor.

A continuación se enumeran las tareas fundamentales que realiza el módulo de comprensión:

1. Comprensión del mensaje recibido.
2. Contextualización del mensaje recibido.
3. Detección de los principales errores gramaticales.
4. Elaboración de las respuestas que el módulo de virtualización ejecutará.

La principal desventaja de este módulo de comprensión es que no ha sido diseñado para entablar una conversación con el usuario. El módulo de comprensión, tras recibir la sentencia transcrita por parte de los reconocedores, detecta posibles errores en la sentencia pero no trata de corregirlos si estos se produjesen. El módulo de comprensión es un sistema que únicamente funciona si las sentencias recibidas son correctas. Por esta razón, el módulo de comprensión no se denominó sistema de diálogo, puesto que no establece un diálogo con el usuario para resolver sentencias erróneas. Simplemente, detecta los errores y los notifica.

Vimos en este punto una limitación a la hora de establecer comunicación con el usuario. Con el módulo de comprensión, la comunicación con el usuario se hace pesada, es un sistema de tipo comando-respuesta,

en el que no hay apenas lugar para el diálogo. Esta fue una de las razones fundamentales por las que surge el trabajo fin de máster presente: generar un sistema de diálogo real que sea capaz no sólo de detectar los posibles errores de las sentencias emitidas por los usuario sino también corregirlos.

1.3.2.1 Diferencias entre el sistema de diálogo y el módulo de comprensión

El sistema de comprensión no realiza todas las tareas que enunciábamos en el apartado anterior. El sistema de diálogo comparte con el módulo de comprensión la tarea de entendimiento del mensaje recibido, la de detección de los errores gramaticales y la de elaboración de la respuesta que el módulo de virtualización ejecutará.

En cambio, la tarea de contextualización no ha sido incorporada al sistema de diálogo al estar fuertemente orientada a la aplicación concreta donde se vaya a utilizar. El módulo de comprensión realizaba bien esta tarea. Si se precisara de su utilización, tan solo deberíamos extraer este submódulo e incorporarlo después al sistema de diálogo.

Hasta aquí hemos hecho una revisión del interfaz de control multimodal del que formaría parte el sistema de diálogo desarrollado en este proyecto fin de máster.

A continuación se dará una breve descripción del sistema de diálogo diseñado.

1.3.3 Breve descripción del trabajo realizado

En este proyecto se describe con detalle la construcción de un sistema de diálogo para un interfaz de control multimodal como el descrito en la sección 1.3.1. La construcción de un sistema de diálogo pasa por varias etapas que describiremos con todo lujo de detalles en el capítulo 3. Sin embargo, no está de mas dar una idea de cuales son dichas etapas y dar una descripción intuitiva de las mismas.

En primer lugar, antes de empezar a desarrollar un sistema de diálogo, tenemos que describir muy bien la aplicación donde se va a utilizar. De esta descripción se generará un listado del vocabulario con el que el sistema de diálogo trabajará. También se debe pensar en las posibles combinaciones de los elementos del vocabulario que se van a permitir y conformar una gramática.

Una vez descritos estos elementos, debemos de pensar en los errores comunes que o bien los usuarios, o bien los sistemas de reconocimiento suelen cometer. En este punto se debe hacer una clasificación de los más frecuentes para que puedan ser detectados y corregidos en pasos futuros.

Después de esta clasificación, se debe pensar en la ejecución que debe tomar el algoritmo para la detección y corrección de los errores anteriores, teniendo siempre presente tanto el vocabulario de la aplicación como la gramática. Es importante tener presente que el sistema de diálogo está orientado a trabajar en un entorno muy concreto, con un vocabulario y gramática muy específico. Pretender construir un sistema de diálogo general, que sirva para todo tipo de entornos está condenado al fracaso al menos en la actualidad. Aprovecharnos de esta restricción permite construir sistemas bastante eficientes.

Una vez hemos determinado la linea de ejecución de nuestro algoritmo, el siguiente paso consiste en implementarlo, estudiando diferentes métodos y técnicas que podemos añadir a nuestro algoritmo. Una de ellas consiste en conferirle al algoritmo memoria, lo cual proporciona al sistema de diálogo una comunicación más humana. Veremos más detalladamente esta técnica en el capítulo 3.

Finalmente, tras comprobar que el sistema de diálogo resuelve los principales errores pidiendo la información pertinente al usuario, podemos añadir algunas técnicas adicionales que complementan al sistema haciendo que sus conversaciones sean más humanas. A modo de ejemplo, nuestro sistema de

diálogo implementa la opción de cancelación del comando anterior, o la emisión de mensajes distintos de solicitud de información. Todo ello enriquece el diálogo del sistema haciéndolo menos robótico.

Una vez hablado de la contextualización y habiendo dado una breve descripción de las partes fundamentales que hemos seguido en la construcción de nuestro sistema de diálogo, vamos a pasar a ver los objetivos generales de este proyecto fin de máster.

1.4 Objetivos

1.4.1 Objetivos generales

El objetivo fundamental del trabajo final es el diseño, implementación y evaluación de un sistema de diálogo que sea capaz de interpretar las sentencias recibidas por parte de los reconocedores del interfaz de control multimodal del que ya hemos hablado. Además, debe garantizar que la sentencias recibidas por parte del usuario sean coherentes con la aplicación que se quiere controlar, incluso tratar de corregir aquellas que no lo sean preguntando al usuario más información, interactuando con él.

1.4.2 Objetivos de diseño

Los objetivos de diseño hacen referencia a los requisitos que se deben cumplir y a los módulos que debemos crear para alcanzarlos en la construcción de nuestro sistema de diálogo:

- Diseño del vocabulario y de la gramática del sistema de diálogo. Mucho del vocabulario y de la gramática es heredado del que utilizaban los reconocedores, sin embargo, el sistema de diálogo puede ser utilizado sin acoplar a la entrada ningún tipo de reconocedor, simplemente introduciendo las sentencias transcritas tal y como el reconocedor las introduciría. Por ello, es posible que el sistema de diálogo contenga algún que otro término diferente del vocabulario propio de los reconocedores.
- Definición de los errores que el sistema de diálogo va a tratar de resolver.
- Diseño del algoritmo de ejecución del sistema de diálogo. El algoritmo de ejecución es la columna vertebral de nuestro sistema de diálogo. Es quien comprueba que la frase introducida por el usuario o por los reconocedores sea correcta y pregunta al usuario para solventarlas en el caso de que sea errónea.
- Diseño de respuestas del sistema de diálogo ante posibles errores cometidos.

1.4.3 Objetivos de implementación

Entre los objetivos de implementación destacamos la generación de varios sistemas:

- Desarrollo de las estructuras de datos que almacenan el vocabulario propio del sistema de diálogo y codifican las reglas de la gramática. Las estructuras deberán independientes de la aplicación que vaya a dar al sistema de diálogo de tal modo que pueda ser utilizado el mismo sistema en otros contextos, simplemente cambiando el vocabulario y las reglas.
- Desarrollo del algoritmo de ejecución en lenguaje de programación $C/C++$ funcionalmente escrito por módulos o al menos en su mayoría. De este modo, se puede llegar a utilizar en otro tipo de aplicación.

1.4.4 Objetivos de evaluación

Entre los objetivos de evaluación tenemos que destacar los siguientes:

- Comprobación de que se cumplen los objetivos de corrección de errores.

1.5 Estructura

El proyecto está estructurado en las siguientes partes:

- **Introducción:** En este capítulo se hará una breve exposición del problema planteado, es decir, del diseño del sistema de diálogo que se pretende crear, contextualizandolo dentro de los proyectos fin de carrera de David[2] y Manuel[1]. Se proponen también en este capítulo la motivación y los objetivos que han impulsado a este proyecto a salir a la luz.
- **Estudio teórico:** Aquí contaremos un breve resumen de los aspectos teóricos relacionados con el sistema de diálogo y haremos una revisión del estado del arte.
- **Desarrollo**: En este capítulo contaremos cada una de las fases que hemos seguido en la construcción de nuestro sistema de diálogo. Se desglosa en varias subsecciones en donde se cuenta el diseño del vocabulario y de la gramática, la definición de los errores a corregir, las estructuras de datos que se utilizan para la búsqueda de información, la creación del algoritmo del sistema de diálogo, el desarrollo de la técnica de cancelación y de generación de respuestas.
- **Resultados:** En este capítulo se muestran los resultados del sistema de diálogo construido.
- **Conclusiones y líneas futuras:** Por último, se hacer un resumen de las conclusiones a las que hemos llegado a lo largo del proyecto y se enumeran algunas propuestas futuras.

Capítulo 2

Estudio teórico

Una buena conversación debe agotar el tema, no a sus interlocutores.

Winston Churchill

2.1 Introducción

En este capítulo se describe la teoría que hemos utilizado a la hora de desarrollar el sistema de diálogo. El capítulo se va a dividir en dos secciones.

- En la primera sección se dará una definición de sistema de diálogo y se hará una clasificación de las partes en las que se descompone.
- En la segunda y última sección se hará una revisión del estado del arte relacionado con los sistemas de diálogo actuales.

2.2 Teoría de los sistemas de diálogo

Los sistemas de diálogo son sistemas que tienen por objeto establecer una conversación con los humanos con una estructura coherente. En general, los sistemas de diálogo pueden emplear texto, voz, imágenes o gráficos, gestos u otros modos para comunicarse en ambos sentido de comunicación (entrada y salida). En este enunciado, hablamos de sistema de diálogo desde un punto de vista general, y desde esta perspectiva, un sistema de diálogo comprende tanto las tareas de reconocimiento como las de síntesis, independientemente del medio de comunicación que empleen en el diálogo. Aquí, tenemos que recordar el capítulo de introducción 1, concretamente la sección 1.1 donde hablamos de los dos puntos de vista desde los cuales podíamos concebir un sistema de diálogo.

Entendido de esta manera, el sistema de diálogo estaría formado por múltiples etapas. El esquema de la figura 2.1 muestra el esquema funcional de un sistema de diálogo genérico.

A continuación se expone cual es la finalidad de cada uno de los siguientes módulos:

- El reconocedor es el primer bloque funcional del sistema de diálogo que hemos esquematizado y que se representa en la figura 2.1. El principal objetivo de este módulo es procesar la señal recibida por parte del usuario y transformarla en una secuencia de palabras reconocidas en formato texto. En

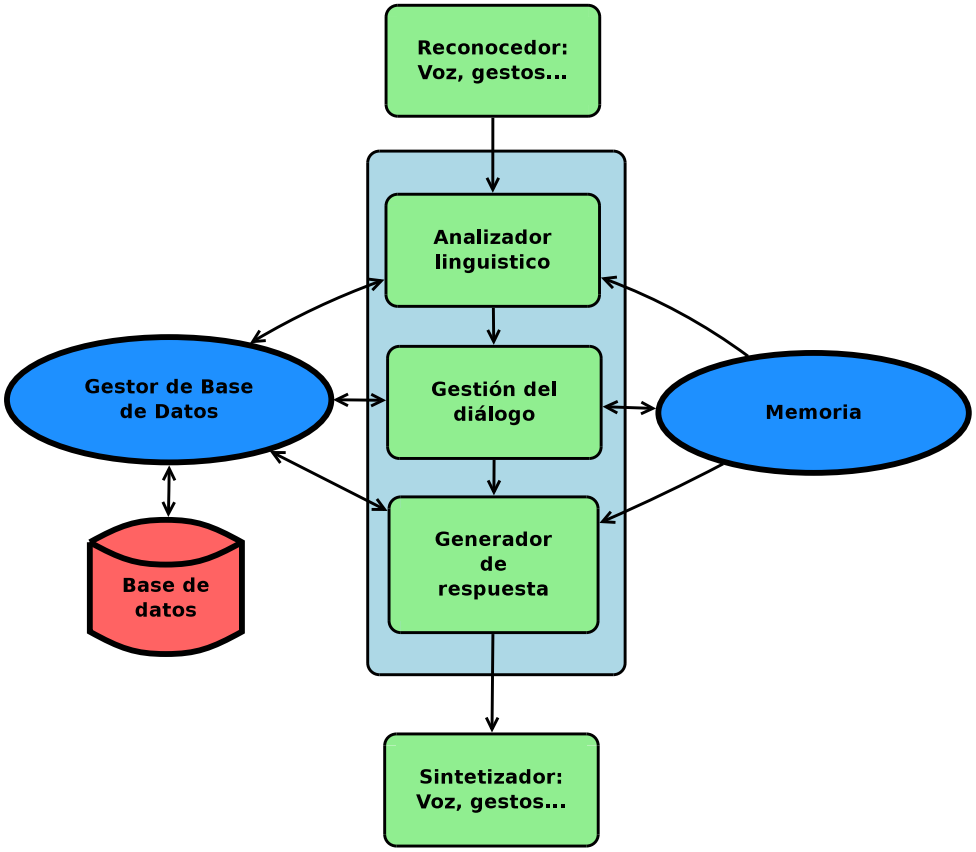


Figura 2.1: Esquema funcional de un sistema de diálogo general.

función del tipo de canal a través del cual el usuario se comunique con el sistema, hablaremos de reconocedores de voz, reconocedores de gestos. . . , pero todos ellos tienen en común que proporcionan a su salida la sentencia transcrita del mensaje de la entrada. El reconocedor está muy ligado con el sensor que recoge la señal hasta tal punto que sus características pueden influir en el buen funcionamiento del mismo. A modo de ejemplo, en la interfaz de control multimodal de la que hablamos en la introducción se construyeron dos reconocedores diferentes, un reconocedor de gestos y un reconocedor de voz. El reconocedor de voz tenía por objeto obtener la transcripción de las sentencias orales emitidas por el usuario, mientras que el de gestos obtiene la trascripción de las "sentencias gestuales" que se definieron en el interfaz de control.

El reconocedor es una de las piezas más importantes de todo el sistema de diálogo. Hasta tal punto alcanza su importancia que, debido a su complejidad, se suele extraer del esquema del sistema de diálogo para ser estudiado por separado. Su estudio da para escribir muchos capítulos de libros y no constituye el objetivo de esta sección. Tan solo queremos hacer una breve referencia a los HMM sobre los cuales se fundamentan la mayoría de los reconocedores que se construyen hoy en día. En el proyecto fin de carrera de Manuel [1] y David [2] se construyeron los reconocedores en base a estos HMM, siguiendo el esquema que se expuso en la figura 1.9. Para más información, consulte la bibliografía de Rabiner [9].

- El segundo bloque del esquema funcional del sistema de diálogo del la figura 2.1 es el analizador lingüístico. El analizador lingüístico toma como entrada la secuencia de palabras transcritas por parte del reconocedor y tiene por objeto obtener la representación semántica de de la sentencia previamente reconocida, es decir, su significado. Este bloque representa el núcleo del sistema de diálogo puesto que traduce las palabras a símbolos con significado. Para nosotros, los símbolos son

representaciones perceptibles de una idea. El significado no es más que el contenido mental que le es dado a estos signos lingüísticos [12]. Esta es precisamente la labor que debe llevar a cabo el sistema de diálogo, obtener el significado del signo. El significado, se representa generalmente por una estructura de datos, como veremos en nuestro propio sistema de diálogo diseñado.

El analizador lingüístico utiliza tanto el vocabulario como la gramática a la hora de obtener el significado. Este módulo es uno de los más importantes del sistema puesto que realiza la labor de abstracción que lleva al sistema a traducir palabras en conceptos.

- El tercer bloque del diagrama funcional de un sistema de diálogo2.1 es el gestor del diálogo. Este módulo tiene por objeto determinar que acción debe realizar el sistema en cada momento. Su finalidad es establecer una interacción lo más cómoda e inteligente posible con el usuario. Para conseguirlo, este módulo suele utilizar varias técnicas como la confirmación de los datos obtenidos, la corrección por medio de interrogación al usuario, la cancelación...

- El módulo de gestión finalmente decide la siguiente acción a efectuar en función de las diferentes opciones que se le presentan. El bloque funcional generador de respuestas (véase 2.1) toma como entrada la decisión del gestor del diálogo y en función de dicha decisión formula una sentencia escrita que servirá como entrada al sintetizador. En la generación de la respuesta, este bloque funcional utiliza técnicas que buscan establecer diálogos lo menos robóticos posibles. Una de entre las múltiples técnicas que existen y que nosotros hemos utilizado en la elaboración de nuestro sistema de diálogo es la aleatoriazación de sentencias equivalentes. Las sentencias equivalentes son sentencias que tienen el mismo significado y buscan en definitiva la no repetición de respuestas, característica muy común en el habla robótica y que deshumaniza el diálogo.

- El sintetizador es el último bloque de entre todos los que se pueden ver en la figura 2.1. El sintetizador traduce la sentencia escrita generada por el bloque anterior al canal de comunicación correspondiente(sonido, gestos,...), para que pueda se entendida por el usuario. A modo de ejemplo, en el caso de la voz, el sintetizador de voz genera la correspondiente sentencia oral a partir de la escrita.

- Si nos fijamos en el diagrama de la figura 2.1, además de todos los bloques funcionales ya explicados, se incluye en el diagrama dos elementos de los cuales el resto del bloques hace uso: El gestor de la memoria y el gestor de la base de datos. A continuación se van a explicar brevemente en consisten estos bloques funcionales:

  - La memoria se encarga de almacenar las representaciones semánticas de cada una de las iteraciones que el algoritmo ejecuta a la hora de resolver las sentencias emitidas por el usuario. Además, almacena las sentencias previamente generadas de tal modo que proporciona un histórico que puede ser utilizado por el resto de módulos en la realización de su tarea. La memoria es un módulo que permite resolver elipsis temporales, contextualizaciones, anáforas...
  - El gestor de la base de datos es quien se encarga de establecer la comunicación con la base de datos. La base de datos es el elemento donde se almacenará la información acerca del vocabulario empleado por el sistema de diálogo o donde se encapsulan las diferentes reglas de la gramática. Mantener una base de datos bien estructurada y ordenada mejora la rapidez con la que el algoritmo del sistema de diálogo se ejecuta. Esto es indispensable sobre todo en sistemas de diálogo que manejan grandes volúmenes de datos en forma de vocabulario y de reglas gramaticales. En todo sistema de diálogo se debe garantizar los tiempos de respuesta para crear un sistema de tiempo real, ya que el usuario debe obtener una respuesta rápida para que no le resulte pesada la conversación.

Hasta aquí hemos visto un esquema de funcionamiento general2.1, una abstracción del funcionamiento de todo sistema de diálogo. Ni que decir tiene que todos los sistemas de diálogo que existen deban seguir al pie de la letra el esquema funcional de la figura 2.1. Algunos de ellos engloban varias funcionalidades en un único bloque funcional, otros, en cambio, desglosan un bloque funcional de los ya vistos en varios de los anteriores.

En la mayor parte de los casos, se considera al sistema de diálogo como un conjunto de bloques como los mostrados en la figura 2.1 pero eliminando el módulo de reconocimiento y el módulo sintetizador. Estos bloques son lo suficientemente complicado como para extraerlos del esquema principal. Además, estrictamente hablando y de acuerdo a la definición que vimos al principio del capítulo, un sistema de diálogo puede actuar siempre que la sentencia que proviene del usuario sea la sentencia transcrita y no una sentencia oral o gestual que deba ser reconocida previamente. Desde este punto de vista, tanto el reconocedor como el sintetizador podrían ser prescindibles.

Una vez dada una definición de sistema de diálogo y explicado el esquema funcional genérico que cumplen todos ellos, en la siguiente sección vamos a hacer una clasificación de los distintos tipos de sistemas que nos podemos encontrar.

2.2.1 Tipos de sistemas de diálogo

Existen varias categorías en las que podemos clasificar los sistemas de diálogo. Muchas de estas categorías se solapan y las distinciones entre unas y otras son confusas. Las más frecuentes son las siguientes:

- Según la modalidad de interacción que use el sistema de diálogo para interactuar con el ser humano, el sistema de diálogo puede ser:

  Basados en texto (*text-based*) La interacción se efectúa a través de la lectura y la escritura de texto plano. No necesita ni reconocedor, ni sintetizador para entablar una comunicación.

  Basados en voz (*spoken dialog system*) La interacción se efectúa a través de las sentencias pronunciadas por el usuario y por el sintetizador de voz.

  Basados en gráficos (*grafical user interface*) La interacción se hace a través de imágenes que el sistema representa por pantalla y por la acción del usuario sobre dispositivos de tipo teclado o ratón sobre las mismas. Desde este punto de vista, son sistemas de diálogo cualquier aplicación gráfica ya que trata de establecer una "conversación" con el usuario.

  Basados en gestos La interacción se realiza a través de gestos.

  Multimodal La interacción puede ser realizada a través de mezcla de las anteriores. El módulo de comprensión del que hablamos en el capítulo de introducción 1 puede encajar dentro de esta clasificación.

- Según la forma en la que se lleve a cabo el diálogo [13]:

  Sistemas de diálogo guiados: Sistemas de diálogo en el que la interacción con el usuario se realiza mediante alternancias cerradas entre pregunta y respuesta. Tiene el inconveniente de que no se puede interrumpir al sistema.

  Sistemas de diálogo cooperativo: Sistema de diálogo basado que no están cerrados, sino que aceptan interrupciones por parte del usuario. Existe un reparto equilibrado del turno de palabra. En este tipo de sistemas se deben de incorporar mecanismos de incoherencias gramaticales.

  Sistemas de diálogo adaptativos Sistemas de diálogo que son capaz de aprender nuevas estrategias comunicativas en función del comportamiento del usuario.

- Según la naturaleza de su construcción, el sistema de diálogo puede ser:

  Sistemas de diálogo de tipo determinista El algoritmo de ejecución del sistema de diálogo es determinista.

  Sistemas de diálogo de tipo estocástico El algoritmo de ejecución del sistema de diálogo recorre una estructura con probabilidades y proporciona la sentencia más probable de entre todas las posibles.

Una vez dado una definición de un sistema de diálogo, habiendo dado un esquema general de los bloques funcionales en los que se descompone y la viendo varias clasificaciones de los mismos, en la siguiente sección se va a hacer una breve revisión del estado del arte en relación con los sistemas de diálogo.

2.3 Estado del arte

La problemática del diseño de sistemas de diálogo es un tema que incluso hoy en día dista mucho de ser resuelta. En general, los sistemas de diálogo han sido un tema muy estudiados y en la última década se han incrementado los esfuerzos por conseguir mejores soluciones al problema. Centrándonos en los sistemas de diálogo vocales, la problemática de estos sistemas heredan los problemas debidos al reconocimiento automático del habla, a la generación de lenguaje natural y a la síntesis de voz.

Pese a todos los esfuerzos empleados en mejorar los resultados de los sistemas de diálogo, hoy por hoy, el esquema básico del mismo sigue siendo una máquina de estados que ejecuta procesos en función de una serie de variables condicionales. La gran mejora que se ha dado en estos últimos años ha sido la modelación del sistema desde un punto de vista probabilístico en vez de seguir el tradicional punto de vista determinista que se venía dando con anterioridad. La consideración de un problema desde el punto de vista trae consigo tomar en consideración todas las posible sentencias recibidas y considerar verdadera la más probable. En este sentido, debemos destacar los trabajos titulados “Towards Building Intelligent Speech Interfaces Through the Use of More Flexible, Robust and Natural Dialogue Management Solutions” [14] de Fernandez-Martinez y compañía, y “Flexible, Robust and Dynamic Dialogue Modeling with a Speech Dialogue Interface for Controlling a Hi-Fi Audio System” [15] también de los mismos autores. En ambos trabajo domina esta alternativa de modelado estadístico del problema, concretamente por medio de redes bayesianas. Centrándonos en las reyes bayesianas y en su aplicación directa como parte del modelado estadístico de un sistema de diálogo sería muy recomendable leer el artículo del mismo autor que mencionábamos en los artículos anteriores y que se titula: “A Bayesian NETWORKS Approach for Dialog Modeling: The Fusion BN” [16].

Relacionadas con la vertiente determinista, aun siendo menos eficaz que la anterior opción, muchos son los trabajos que se han desarrollado. Entre ellos, queremos destacar el trabajo de Ferreiros, y Colas titulado: “Controlling a HIFI with a Continuous Speech Understanding System”, sistema de diálogo que trata de controlar un dispositivo HIFI por medio de un sistema de diálogo de tipo continuo”.

Con el objetivo de proponer un ejemplo de sistema de diálogo completo, se propone el proyecto titulado “A speech interface for air traffic control terminals” cuyo objetivo principal imponía la construcción de un interfaz de diálogo oral con el objetivo de controlar el tráfico aéreo que se produce en un aeropuerto.

2.4 Conclusiones

En este capítulo, hemos dado un breve repaso a la teoría de los sistemas de diálogo. En el capítulo se ha expuesto una breve definición general de sistema de diálogo. Después, se ha proporcionado un esquema funcional general, representado en la figura 2.1, que describe a un sistema de diálogo genérico. Tras este esquema, se han expuesto varias clasificaciones de sistemas de diálogo diferentes, y por último, se ha proporcionado una breve descripción del estado del arte.

Capítulo 3

Desarrollo

Para dialogar, preguntad primero; después... escuchad.

Antonio Machado

3.1 Introducción

En este capítulo se explica tanto la implementación como los desarrollos llevados a cabo a lo largo del proyecto. Se ha dividido el capítulo en una serie de apartados cada uno de los cuales se explicarán de forma independiente, facilitando de esta manera tanto la comprensión como la reutilización de los mismos en trabajos futuros.

Los apartado en los que hemos dividido este trabajo no coinciden de forma exacta con los bloques funcionales propios a todo sistema de diálogo, tal y como lo describimos en el capítulo 2. Algunos secciones de este capítulo constituyen submódulos de los bloques funcionales explicados en teoría. Por ello, para establecer una correspondencia entre los puntos que se expondrán a continuación y los módulos explicados en la teoría se ha desarrollado el siguientes esquema. Antes de empezar a numerarlas, le recomendamos revisar el diagrama funcional que describía el funcionamiento genérico de un sistema de diálogo y que se muestra en la figura 2.1 de la página 14.

El capítulo se compone de los siguientes apartados:

Definición del vocabulario: En este punto se describe el vocabulario empleado por el sistema de diálogo. Se corresponde con el bloque funcional que llamamos analizador lingüístico en teoría. Además, el conjunto de vocabulario se almacena el base de datos que vimos.

Definición de la gramática: En este apartado se explicarán las reglas que rigen las posibles combinaciones de palabras que pueden darse, reglas fundamentales sobre las que el sistema de diálogo se basa para analizar las sentencias de los usuarios y generar respuestas. También se corresponde con el bloque funcional que llamamos analizador lingüístico en teoría.La gramática también se encapsula en la base de datos que vimos en la figura 2.1 de la página 14.

Definición y clasificación de errores: Este punto se centrará en hacer una clasificación de los errores gramaticales que el algoritmo tratará de resolver. Este bloque realiza funciones también del analizador lingüístico.

Estructura de datos: En este punto se describirá la organización de la gramática en una estructura de datos de fácil recorrido. Este módulo es característico del gestor de la base de datos, así como de la base de datos en sí.

Algoritmo de búsqueda: El algoritmo de búsqueda describe la manera en la que el sistema de diálogo resuelve cada una de las sentencias de entrada. Este apartado incluye el diagrama de flujo de ejecución del propio algoritmo. Este bloque se corresponde con el gestor del diálogo. Engloba también al bloque de memoria.

Técnica de cancelación: En esta sección se explica la técnica de cancelación diseñada e incorporada a nuestro sistema de diálogo. Este bloque es un submódulo del gestor de diálogo.

El módulo generador de respuestas: Este módulo explica las técnicas empleadas por el sistema de diálogo a la hora de generar una respuesta hacia el usuario. Se corresponde íntegramente con el generador de respuestas.

3.2 Definición del vocabulario

El primer paso que debemos realizar a la hora de construir un sistema de diálogo completo es definir tanto el vocabulario que empleara el sistema como las posibles combinaciones que se van a dar entre los elementos del mismo, atendiendo a un conjunto de reglas o gramática. Restringir el vocabulario que vamos a utilizar para nuestra aplicación, así como las posibles combinaciones que se pueden dar entre los elementos del mismo, facilita la labor de construcción del sistema de diálogo. Sin embargo, un vocabulario reducido junto con una gramática muy estricta da lugar a sistemas de diálogo deshumanizado, con un comportamiento robótico palpable. La regla que debemos aplicar es una solución de compromiso: *Se debe escoger un vocabulario y una gramática lo suficientemente estricta para desarrollar un sistema de diálogo sencillo y conciso para la aplicación en la que se ejecutara, pero sin olvidar introducir cierto grado de abstracción tanto en el vocabulario como en las combinaciones de frases que un usuario puede hacer para evitar desarrollar un sistema de diálogo robótico o deshumanizado.*

En nuestro caso particular, el vocabulario que debemos utilizar para nuestro sistema de diálogo viene definido por el propio interfaz hombre-máquina que desde el principio se quiso implementar (descrito en el capítulo 1). Nuestro proyecto fin de máster es una pieza clave de un interfaz de comunicación hombre-máquina muy concreto. Recordemos que el interfaz pretende controlar ciertos objetos de un espacio inteligente situados en la sala *geintra-ispace*(mostrada en la figura 1.10, en la página 8). Cada objeto llevará asociadas una o varias acciones que se pueden realizar sobre dicho objeto o que dicho objeto puede realizar. Es el caso del objeto `puerta`, que lleva asociada las acciones `abrir` o `cerrar` pero no la acción `enciende`, véase la figura 3.1. Los objetos que podemos controlar son los actuadores del espacio inteligente que vimos en el capítulo 1 (a partir de la página 1) de introducción y que recordamos en la figura 1.5.

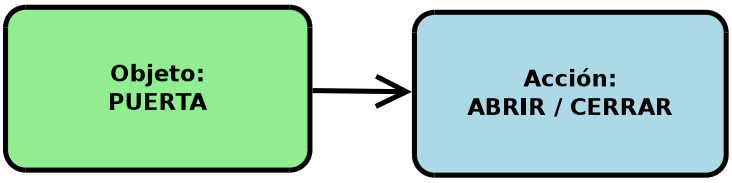


Figura 3.1: Acciones relacionadas con el objeto `puerta`.

El apéndice B (*Vocabulario y gramática del sistema de diálogo*), contiene todo el vocabulario del sistema de diálogo. Además del vocabulario, se incluye también todas las reglas de la gramática, reglas

que se explicarán en la sección 3.3. Este apéndice está organizado en función de si las palabras del vocabulario son "acciones" o "objetos", clasificación que se verá más en detalle en la sección 3.3. Tan solo queremos referenciar este apéndice para mostrar el vocabulario que el sistema de diálogo utilizará.

3.3 Definición de la gramática

La gramática define por sí misma todas las posibles sentencias que un usuario puede generar. La gramática recoge todas las combinaciones posibles entre los elementos del vocabulario del que dispongamos, es decir, aglutina todas y cada una de las posibles combinaciones entre los objetos y las acciones asociadas a los mismos. El vocabulario por sí mismo no constituye ninguna orden o sentencia per se. Las sentencias se conforman por medio de una acción y un objeto sobre el que recae la acción. Para nuestro caso particular, debemos recordar que el sistema de diálogo es la pieza del interfaz de control multimodal que se se ejecuta tras el reconocedor(de gestos, de voz o ambos) y cuyo resultado ejecutara el módulo que simula el espacio inteligente real (que se describe en el capítulo 1). Las sentencias emitidas por el usuario ejercerán una influencia sobre los actuadores (véase la figura 1.5 en la página 6) del espacio inteligente que clasificaremos en dos tipos:

- Los actuadores estáticos: Modificaremos sus estados por medio de reglas simples basadas en acciones concretas.
- Los actuadores dinámicos: Modificaremos el movimiento del único actuador dinámico que tendremos en la sala, el robot, a partir de acciones concretas.

Para la construcción de la gramática, se ha realizado una clasificación de cada una de las palabras que el reconocedor proporciona a su salida. La clasificación se ha realizado con criterios gramaticales y podemos distinguir tres tipos de elementos diferentes:

Una acción Una acción es un comando que implica la realización de un cambio en algún objeto del espacio inteligente. Ejemplos de acciones son `abrir`, `cerrar` o `enciende`.

Un objeto Un objeto hace referencia a un elemento del espacio inteligente sobre el que se va a ejecutar una determinada acción. Ejemplos de objetos son `puerta`, `ventana` o `persiana`.

Un modificador Un modificador es una palabra que sirve de soporte en la sentencia. Se puede utilizar un modificador para especificar un objeto del espacio inteligente de forma unívoca `SWITCH OFF THE RIGHT LAMP` o para modificar de alguna manera la ejecución de una determinada acción `MORE LIGHT`.

A la hora de construir la gramática del sistema de reconocimiento se pueden pensar en múltiples reglas que hagan muy flexible a nuestro sistema. La complejidad de la gramática redunda en la flexibilidad del diálogo que con posterioridad el usuario tendrá con el sistema. Una gramática muy simple, con reglas muy estrictas, da lugar a sistemas de diálogo sencillos de implementar pero con conversaciones muy robóticas. En cambio, una gramática más compleja que permite múltiples formas de expresar la misma idea aumenta la sensación de inteligencia de la conversación. En primera instancia es conveniente empezar a construir una gramática simple donde en cada sentencia aparezca un único comando a ejecutar, es decir, con una única acción. La gramática simple que se ha elaborado impone al usuario que pronuncie o gesticule en primer lugar el objeto sobre el que recaerá la acción y posteriormente la acción que se quiere realizar sobre él, o viceversa. El siguiente esquema muestra de forma simplificada la gramática simple que se ha desarrollado y que se muestra en la figura 3.2.

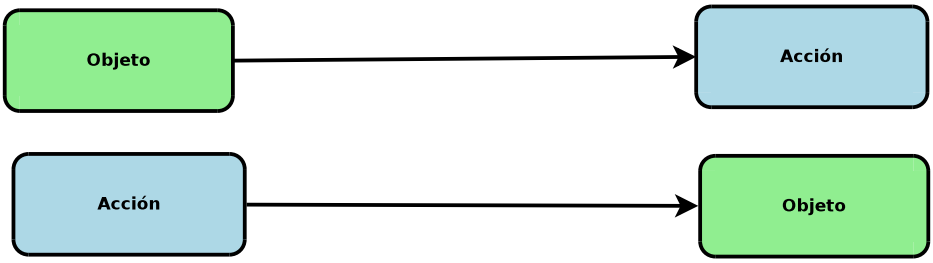


Figura 3.2: Estructura de la gramática de gestos.

La gramática descrita por el diagrama de flujo de la figura 3.2 recoge el diálogo que un usuario ideal tendría dentro del sistema, entendiendo como un usuario ideal aquel que conoce cómo combinar los gestos de acuerdo a la definiciones que recoge dicha gramática. La gramática descrita en en la figura 3.2 no incluye ningún tipo de modificador. Esto es debido a que la ejecución de los comandos para esta aplicación no requiere de forma explicita que se mencione ningún modificador. Los comandos poseen sentido propio siempre que las sentencias posean un objeto y una acción relacionada con dicho objeto. Pese a todo, si la sentencia incluyese algún modificador, el mismo sería tratado como un error o una excepción puesto que en la gramática no están contemplados. En el siguiente nivel de complejidad de nuestra gramática podría incluirse los modificadores tal y como se muestra en la figura 3.3.

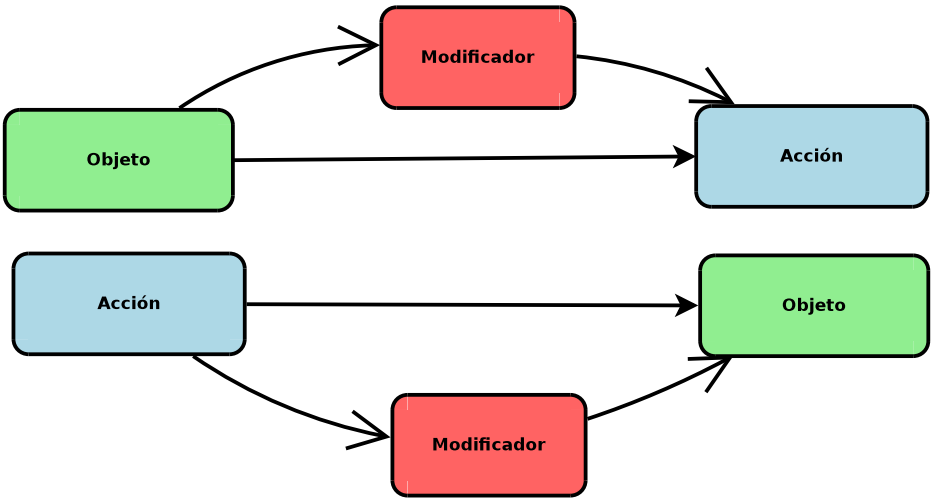


Figura 3.3: Definición de una gramática simple incluyendo modificadores.

A modo de ejemplo, las reglas de la gramática que hemos definido espera estructuras como las que se muestran en la figura 3.3, pero no espera recibir una sentencia del estilo `objeto` sin ser acompañada de la `acción`. Podríamos llegar a querer que el interfaz encendiera todas las lámparas cuando el usuario gesticulara `LÁMPARA`, sin que el usuario gesticulara `ENCIENDE`. Una gramática pulcra o estricta no permitiría tales concesiones, en cambio una gramática más laxa si que trataría de acercarse más a los deseos de los usuarios, desarrollando diálogos mas humanos.

Tal y como queda definida la gramática por la gráfica de la figura 3.3, el usuario no podría decir dos comandos seguidos en la misma sentencia. A modo de ejemplo, sentencias del tipo `ABRE LA PUERTE Y ENVIENDE LA TELEVISION` no están permitidas. La adición de esta funcionalidad por medio de conjunciones copulativas `Y / AND` aporta flexibilidad y genera el sentimiento de conversación humana. La gramática simple desarrollada tampoco incluye la posibilidad de enumerar una serie de objetos sobre los que se aplicará la misma acción. Sentencias como `ABRE LA PUERTA, LA VENTANA y LAS PERSIANAS` no se permiten. Por esta razón se hace conveniente aumentar la complejidad de la estruc-

tura de la gramática permitiendo que existan varios comandos concatenados en la misma sentencia. En la figura 3.4 se representa un diagrama de flujo de una gramática más compleja que incluye todos estos casos.

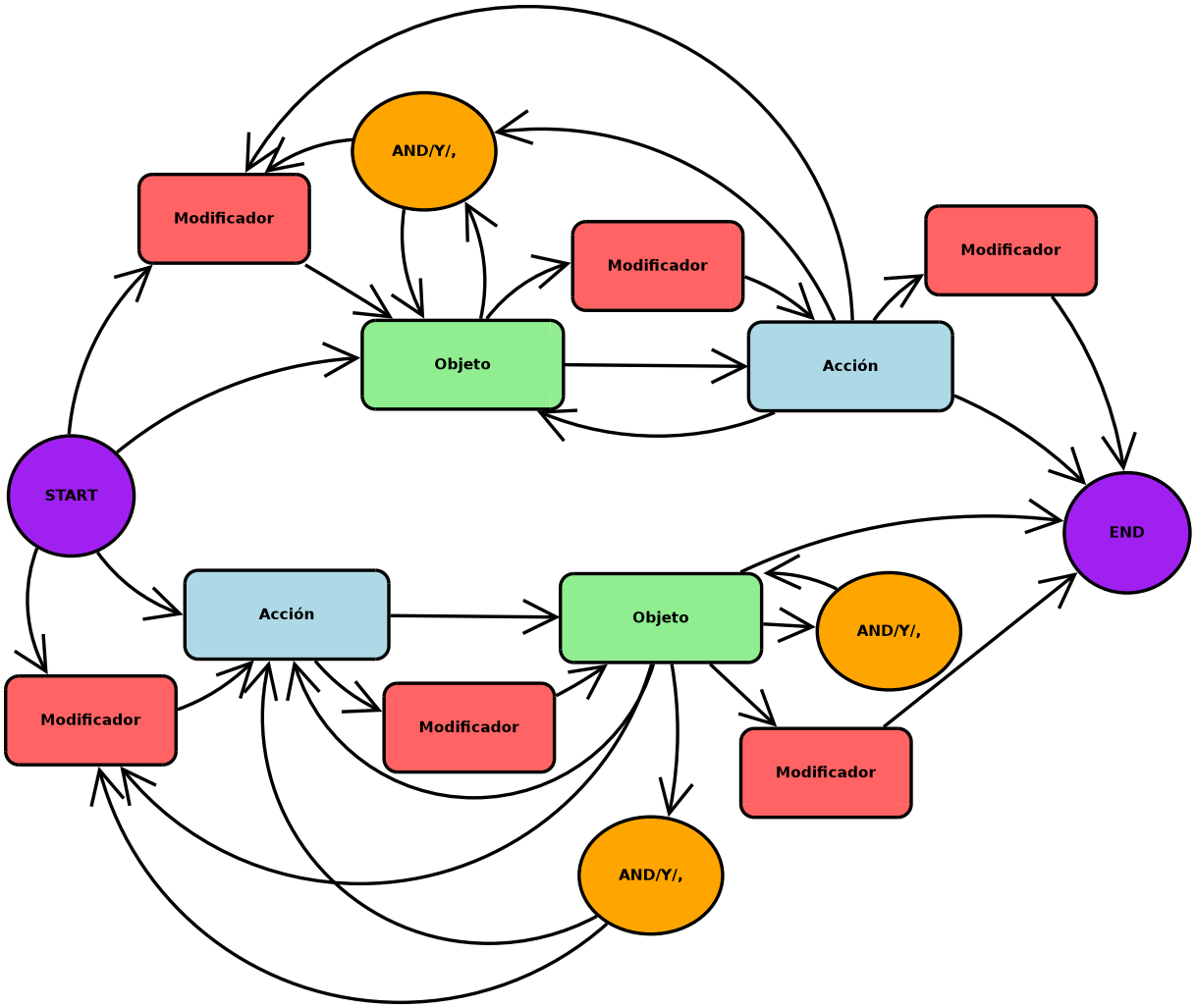


Figura 3.4: Definición de una gramática compleja.

El apéndice B (*Vocabulario y gramática del sistema de diálogo*), contiene tanto el vocabulario como las reglas de la gramática que nuestro sistema de diálogo utiliza.. Este apéndice organiza el vocabulario y la gramática mostrando en primer lugar las acciones, y exponiendo a continuación los objetos relacionados con ellas.

En general, cualquier sentencia emitida por el usuario que no cumpla con la gramática es errónea y como tal debe de ser tratada. El sistema de diálogo tenga que detectar estas incoherencias que existen entre la sentencia emitida y la sentencia prototipo y manejar los distintos tipos de errores que se puedan producir. En la siguiente sección se explicarán los errores más comunes que el sistema de diálogo debe manejar. A modo de ejemplo, entre los mas comunes están las elipsis, las repeticiones, errores de contextualizaciones, etc.

3.4 Definición y clasificación de errores

Se dice que se ha cometido un error en la sentencia cuando la sentencia por si misma carece de sentido por si misma o dispone de un significado parcial. Existen diversos tipos de errores que un usuario o un sistema

de reconocimiento puede cometer a la hora de emitir una sentencia y dependen fuertemente del sistema diseñado. Una sentencia que por si misma no tiene un significado específico para un sistema dado, puede contender sentido completo para otro. A modo de ejemplo, un usuario que emite la sentencia `CIERRA` en un espacio inteligente donde solamente existe un único objeto que se pueda cerrar tiene sentido completo, en cambio, si en el espacio inteligente existen varios elementos sensibles a esta acción, tal vez, sin otro tipo de información, la sentencia carezca de sentido y se haya producido un error.

Entre los múltiples tipo de error que nos podemos encontrar se detectarán y se tratará de corregir los errores gramaticales y los errores semánticos. Por supuesto, también existen otros tipos de errores que no trataremos en nuestro sistema. Ejemplo de ellos pueden ser los errores contextuales.

3.4.1 Errores gramaticales

Se comete un error de tipo gramatical cuando la sentencia emitida por el usuario o bien la sentencia reconocida por el propio sistema de reconocimiento no cumple las reglas recogidas en la gramática. Un error de tipo gramatical depende fuertemente de las normas que rigen la combinación del vocabulario descritas en la gramática.

Se han realizado dos tipos de clasificaciones diferentes de los errores gramaticales que se pueden cometer. La primera distingue los errores en base a la forma en la que se alejan las sentencias del prototipo estándar; la segunda, clasifica los errores en función de la complejidad y del número de fallos detectados. En este punto es fundamental recordar cual es la secuencia básica que se definió en el apartado 3.3. El esquema básico de generación de una sentencia viene definido por la figura 3.5.

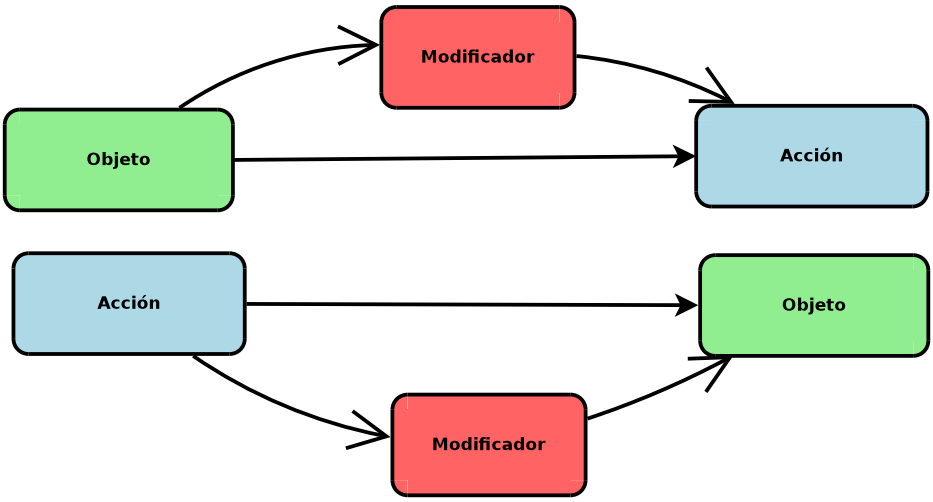


Figura 3.5: Estructura de la gramática.

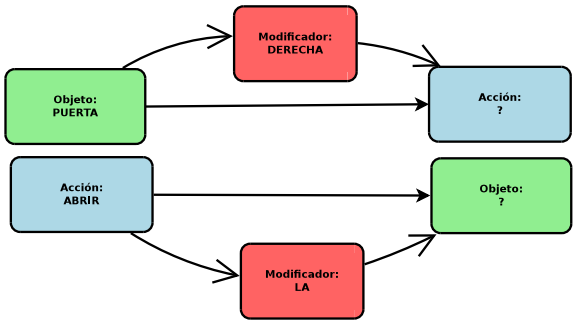
La primera clasificación de errores distingue los mismos en función del modo en que alejan del prototipo ideal de sentencia inteligible. De este modo, podemos encontrar los siguientes errores:

Error por omisión: Sucede cuando uno de los elementos de la sentencia emitida por el usuario o reconocida carece de algún elemento que se considera obligatorio. En nuestro caso particular, se espera recibir en base al ideal de sentencia (véase figura 3.5) un objeto y una acción que ejecutar sobre él. Si alguno de estos dos elementos que el sistema de diálogo requiere no están presentes, la sentencia posee un error por omisión. Véase la figura 3.6a

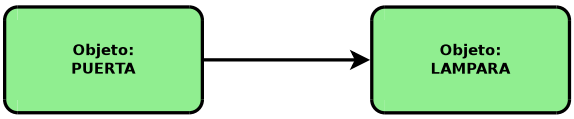
Error por sustitución: Sucede cuando la sentencia contiene todos los elementos de la sentencia estándar (véase la figura 3.5) pero alguno de sus elementos ha sido sustituido por otro que bien con la

misma función gramatical, bien con una función gramatical diferente. Pongamos por caso el ejemplo de la figura 3.6b. Se observa como la sentencia contiene el objeto `PUERTA` y el objeto `LAMPARA`, pero ninguna acción. El elemento `LAMPARA` ha sustituido a la acción `ABRIR` y por lo tanto se ha producido un error por sustitución.

Error por inserción: Se produce en aquellas sentencias que poseen más elementos de los necesarios, desde el punto de vista de sentencia ideal, para que tenga sentido por sí misma. En la figura 3.6c se muestra un claro ejemplo de este tipo de error. La sentencia dispone del objeto `ABRIR` y va seguida de la acción `PUERTA` que de por si tiene significado completo. Sin embargo, la sentencia tiene un error por adición ya que después del objeto `PUERTA` le sigue la acción `CERRAR`. La acción `CERRAR` se añade a la sentencia y deja de tener sentido.



(a) Error por omisión



(b) Error por sustitución

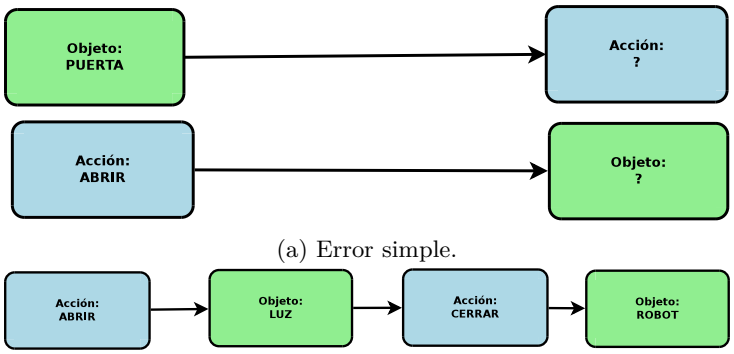
(c) Error por adición

Figura 3.6: Errores gramaticales en funcion de la forma en que se alejan de la sentencia ideal

En la segunda clasificación, nos centraremos en la complejidad del error así como en el número de errores presentes. Una sentencia podrá presentar errores simples o complejos como los que se muestran en la figura 3.7. La definición de estos tipos de errores se describe a continuación:

Error simple: Las sentencias con un error simple, son sentencias en las que falta alguno de los elementos que definen el prototipo estándar de sentencia (véase la figura 3.5). Los errores simples suelen ser errores por omisión en sentencias con una única acción, de ahí su denominación de simples. En la figura 3.7a pueden verse ejemplos de errores simples. En general, los errores simples son debidos a olvidos por parte del usuario emisor de la sentencia.

Errores complejos: Sucede en sentencias compuestas, en las que están implícitos más de un comando incompleto. Ejemplo de ellos son sentencias como las que se muestran en la figura 3.7b. Los errores complejos requieren varias iteraciones del algoritmo ejecutado por el sistema de diálogo para resolver todas las incongruencias encontradas. En el ejemplo propuesto, puede verse como la sentencia emitida posee dos acciones que no se corresponden con los objetos sobre los que se aplica. El algoritmo ejecutado por el sistema de diálogo deberá resolver este tipo de errores y no existe una única forma de solucionarlos.



(a) Error simple.

(b) Error complejo.

Figura 3.7: Errores gramaticales en función de la complejidad.

3.4.2 Errores semánticos

Los errores semánticos son errores de concepto. Las sentencias con errores conceptuales pueden cumplir la sentencia básica que se mostró en la figura 3.5 pero carecer de sentido completo. Gramaticalmente, la regla fundamental que la gramática define exige que en toda sentencia exista una acción y un objeto. Puede suceder que el objeto sobre el que se aplica la acción no estén relacionados. Se produce en este caso un error de tipo semántico. La figura 3.8 muestra un ejemplo de este tipo de error. En dicho ejemplo, la acción `ABRIR` no concuerda semánticamente con el objeto `LUZ`. Con el objeto `LUZ` caben otro tipo de acciones como `ENCENDER` o `APAGAR`.

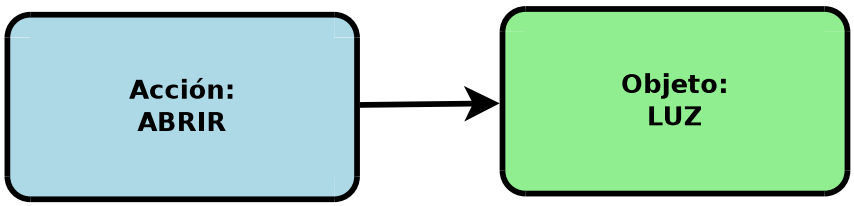


Figura 3.8: Error semántico.

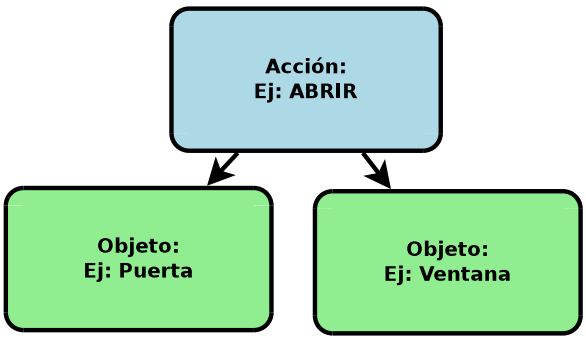
3.5 Estructura de datos

Hablaremos en esta sección de como nuestro sistema de diálogo estructura los datos para que la búsqueda por los mismos sea lo mas eficientemente posible.

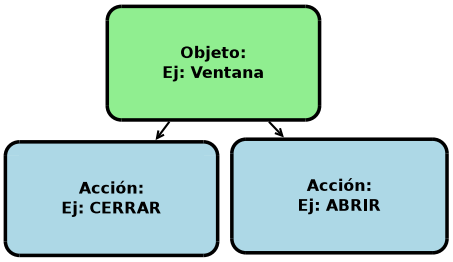
En primer lugar debemos de hablar de la organización de los datos. El sistema de diálogo debe conocer tanto el vocabulario como las principales reglas que definen las sentencias que son correctas. Como vimos en la sección 3.3, en la imagen de la figura 3.2 se puede apreciar como son las sentencias simples recibidas en el sistema de diálogo. Las sentencias simples se forman a partir de una acción y un objeto relacionado con ella o viceversa. Pero, en nuestro caso particular ocurre que el sistema de diálogo puede recibir sentencias del tipo acción junto con más de un objeto relacionado, véase la figura 3.4. Ejemplos de este tipo de acción son `ABRE LA PUERTA y LA VENTANA`. Este tipo de implicación no se da en sentido inverso, es decir, no existen frases en las que un objeto vaya acompañado por varias acciones, al menos, en nuestra aplicación particular. A modo de ejemplo, no existen sentencias del tipo `PUERTA ABRE CIERRA`.

A la hora de organizar la información, resulta conveniente aprovecharnos de dicha información. Es más fácil organizar la información como arboles cuyo raíz sea una determinada acción (véase la figura 3.9a)

y del que cuelguen todos objetos que estén relacionados con ella que al revés (véase la figura 3.9b). De esta forma tendremos menos arboles que recorrer, más hojas en cada uno de los arboles. Frases como la que mostramos en el párrafo anterior se recorren con mayor facilidad con este tipo de estructura. En la figura 3.9, se muestra esta idea.



(a) Árbol por acciones.



(b) Árbol por objetos.

Figura 3.9: Posibles estructuras de organización de los datos.

3.5.1 Los ficheros de almacenamiento

La estructura en árbol con raíz cada una de las acciones y con objetos relacionados se debe almacenar en una estructura de datos que sea de fácil diseño y permanente en el disco. Por ello se ha elaborado una serie de ficheros que almacenarán toda la información. A continuación se describen los ficheros que son necesarios crear para el correcto funcionamiento del sistema de diálogo:

- El fichero "listaDeAcciones.list" almacena en cada línea el número identificador de la acción y el nombre que representa la acción. Normalmente, la acción se representa con la transcripción en inglés. Por ejemplo, la acción `abrir` se representa con la palabra ingles `OPEN`. De cada una de estas acciones representadas por su número de identificación y el nombre de la acción se generan tres fichero con los siguientes nombres: "nombreAccion.action", "nombreAccion.object", "nombreAccion.modifier". A continuación se explican que almacena cada uno de estos ficheros.

- El fichero "nombreAcción.action" almacena en cada linea las posibles transcripciones que la acción puede tener. Por ejemplo, la acción `abrir` identificada por el nombre `OPEN` y con numero identificador de la acción 10 tiene las siguientes como transcripciones posibles `OPEN` y `ABRE`. Conceptualmente, la acción es la misma pero la transcripción es totalmente distinta. Esta es la manera que permite al sistema de diálogo abstraerse al mundo de los conceptos y no quedarse en las palabras. La abstracción era uno de los requisitos que pedíamos y con este mecanismo se consigue.

- El fichero "nombreAcción.object" guarda todos los objetos asociados a la acción indicada en el nombre del fichero. También guarda el identificador del objeto concreto. Por ejemplo, el fichero

"open.object" contiene los siguientes objetos asociados a la acción `open`: `DOOR`, `PUERTA`, `VENTANA`, `WINDOWS`. En este caso, `DOOR` y `PUERTA` tienen el mismo identificador, el 0, puesto que hacen referencia al mismo objeto, mientras que `VENTANA` y `WINDOWS` tienen otro identificador distinto, el 1. De esta manera, podemos asociar palabras que hacen referencia a los mismos objetos proporcionando el grado de abstracción deseado.

- El fichero "nombreAcción.modifier" guarda los modificadores asociados a la acción indicada en el nombre del fichero. Además, guarda el identificador del modificador concreto. Hace las veces del fichero anterior pero guardando los datos relacionados con los modificadores.

El algoritmo del sistema de diálogo no consulta las estructuras gramaticales directamente de los ficheros de almacenamiento, sino que lo carga en memoria. El volumen de datos que ocupan las estructuras en árbol de nuestra aplicación, justifica su carga en memoria. Las estructuras de datos que veíamos en la figura 3.9 son leídas de los ficheros donde están almacenadas físicamente y son almacenadas en estructuras para que el algoritmo del sistema de diálogo pueda leerlas fácilmente sin tener que hacer llamadas al sistema.

3.5.2 Las estructuras de carga en memoria

Una lista de objetos de la clase de $C++$ definida en figura 3.10 almacena la misma información que los fichero en el disco. La estructura ha sido diseñada en lenguaje $C++$ debido a la disposición de contenedores previamente programados como lo es la clase `vector`.

```
1  class c_Action{
2  public:
3    int actionID;                          // Identificador de Acción
4    vector<int> objectID;                  // Identificadores de Objetos asociados
5    vector<int> modifierID;                // Identificadores de Modificadores
         asociados
6
7    string actionFilename;                 // Nombre del fichero de la acción
8    string objectsFilename;                // Nombre del fichero de los objetos
9    string modifiersFilename;              // Nombre del fichero de los
         modificadores
10
11   vector<string> actionEntry;            // Acciones conceptualmente equivalentes
12   vector<string> objectEntry;            // Objetos asociados
13   vector<string> stringmodifierEntry;    // Modificadores asociados
14
15   //Constructores
16   c_Action();                            // Constructor
17   c_Action(int ID, string action_file);  // Destructor
18 };
```

Figura 3.10: Definición de la clase en $C++$ donde se cargan las reglas gramaticales y el vocabulario de la aplicación

El algoritmo del sistema diálogo que se explicará en la sección 3.6, recorre un vector de objetos como el de la figura 3.10 a la hora de encontrar acciones, objetos o modificadores en las sentencias que recibe como entrada. Si analizando una sentencia de entrada encuentra alguna coincidencia, el propio algoritmo generará una lista donde almacena los elementos comprendidos. La lista ha sido creada en lenguaje C de programación por retrocompatibilidad con los proyectos de David[2] y de Manuel[1]. En la figura 3.11 se muestra tanto la cabecera de la lista como los elementos que contiene. Esta lista tiene las siguientes características:

```
typedef struct TUnderstoodAction{
  int actionID;              // Identificador de la acción
  char *actionWord;          // nombre de la acción encontrada

  int numObjects;            // Numero de objetos relacionados
  int *objectIDs;            // Identificadores de los objetos relacionados
  char **objectWords;        // Nombre de los objetos

  int numModifiers;          // Número de modificadores
  int *modifierIDs;          // Identificador de los objetos relacionados
  char **modifierWords;      // Nombre de los modificadores
} TUnderstoodAction;


typedef struct TUnderstoodSentence{
  int numActions;                          // Número de acciones encontradas
  TUnderstoodAction *understoodAction;     // Puntero al primer elmento
} TUnderstoodSentence;
```

Figura 3.11: Definición de las estructuras TUnderstoodSentence y TUnderstoodAction

- La cabecera es una estructura de datos de tipo *TUnderstoodSentence*. Por cada acción encontrada en la sentencia, se incrementa el contador de las acciones *numActions* en una unidad y se añade una estructura de datos *TUnderstoodAction*.
- Los elementos de la lista son de tipo *TUnderstoodAction*. En esta estructura se almacena toda la información de cada acción encontrada en la sentencia de entrada, junto con los objetos encontrados.

En la siguiente sección veremos el algoritmo de ejecución del sistema de diálogo. En ella se explicarán como el algoritmo utiliza las estructuras de datos que hemos visto en esta sección.

3.6 Algoritmo de ejecución

En este punto, describiremos como el algoritmo de actuación que ejecución el sistema de diálogo. Como punto de partida se ha elaborado un gráfico resumen simplificado que contextualiza la labor del sistema de diálogo entre los bloques funcionales del interfaz de control multimodal. La imagen de la figura 3.12 muestra este gráfico.

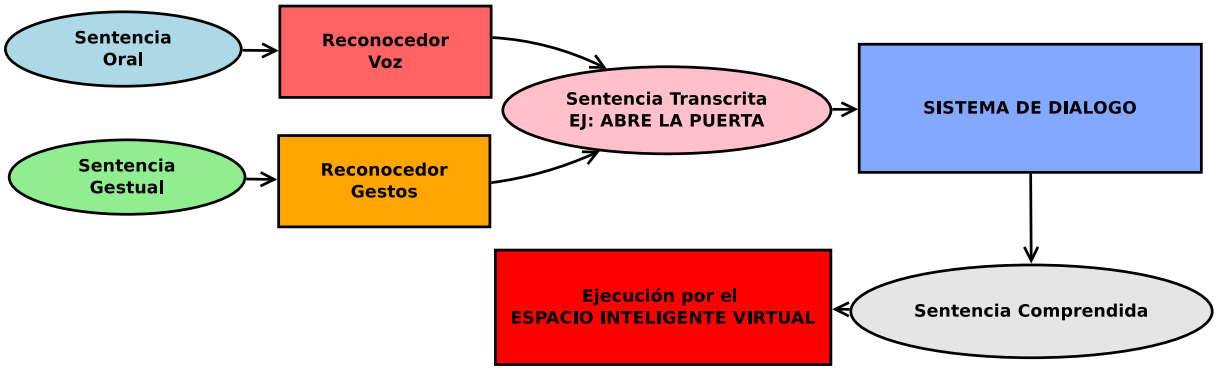


Figura 3.12: Esquema funcional simplificado del sistema de diálogo.

El esquema de la figura 3.12 muestra como el sistema de diálogo toma como entradas las sentencias reconocidas por los reconocedores de voz y de gestos. El reconocedor de voz tiene como objetivo encontrar la transcripción escrita de las sentencias orales que los usuarios del espacio inteligente pronuncie, mientras que el reconocedor de gestos tiene por objeto encontrar las transcripciones de las sentencias gestuales

como se puede ver en el esquema de la figura 3.12. Una vez obtenida la sentencia transcrita por parte de los reconocedores, es el momento de actuar para el sistema de diálogo. El sistema de diálogo tiene como principal objetivo proporcionar a su salida sentencias comprendidas, interpretables, que puedan ser ejecutadas por el modulo que virtualiza el espacio inteligente. El esquema de la figura 3.12 es solo una simplificación funcional para contextualizar la labor del sistema de diálogo con respecto al resto de los módulos funcionales. El esquema muestra el caso particular en el que el sistema de diálogo recibe una sentencia correcta, la comprende y proporciona a su salida una sentencia correcta. Un esquema general no sigue estos pasos tan concretos. Es verdad que la probabilidad de recibir una sentencia correcta será elevada (o se supondrá que es elevada), sin embargo es posible que, o bien el usuario, o bien el reconocedor fallen en algún momento determinado y por lo tanto, el sistema de diálogo, al detectar dicho fallo, deba llamar al sistema de diálogo para requerir al usuario más información que lo relacione. Un esquema de de funcionamiento del sistema de diálogo mas generalista se muestra en la figura 3.13.

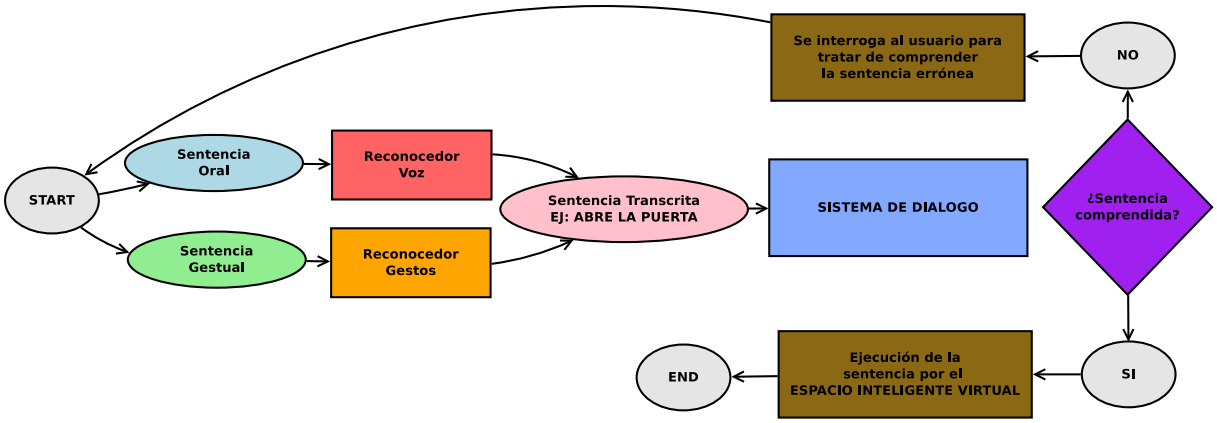


Figura 3.13: Esquema funcional complejo del sistema de diálogo.

Como se observa en la figura 3.13, el sistema de diálogo puede requerir del usuario que aporta mas información a la sentencia para ser comprendida si en ella se detecta un error. En la figura esto se ve reflejado por medio del bucle de realimentación principal del sistema de diálogo. Este lazo de realimentación constituye el núcleo principal del algoritmo de ejecución del sistema de diálogo. En la figura 3.14 se muestra el algoritmo de ejecución simplificado, caracterizado por dicho bucle infinito.

El algoritmo que ejecuta el sistema de diálogo tiene como objetivo final comprender las sentencias transcritas que recibe. Las sentencias que el sistema recibe pueden tener o no errores de diversos tipos, recuerde la sección 3.4. Una de las labores que deberá llevar a cabo el algoritmo será la detección de dichos errores. Si no se detecta ningún tipo de error, el sistema de diálogo dará luz verde al modulo que virtualiza el espacio inteligente para que pueda ejecutar el correspondiente comando. Si un error se ha detectado, se debe proceder a preguntar al usuario por el error cometido para conseguir solucionarlo. Así, sumando ambas informaciones, el sistema podrá comprender y ejecutar en consecuencia la información recibida.

En la mayoría de los casos, las sentencias erróneas no poseen una configuración aleatoria fuera de contexto que hagan inevitablemente incomprensible la frase en el contexto que nuestro espacio inteligente define. Es decir, las sentencia erróneas no son comprendidas por el sistema porque sus errores de base sean sentencias con un vocabulario ajeno al sistema, o construcciones gramaticales excesivamente complicada. En la mayor parte de los casos, los errores se producen por omisiones de palabras que deberían aparecer en la sentencia para su correcta interpretación, o por incoherencias entre palabras que no concuerdan semánticamente. Estas sentencias errónea contienen información parcial comprensible para el sistema de diálogo con lo que puede almacenarse en algún tipo de estructura con la esperanza de que pueda ser

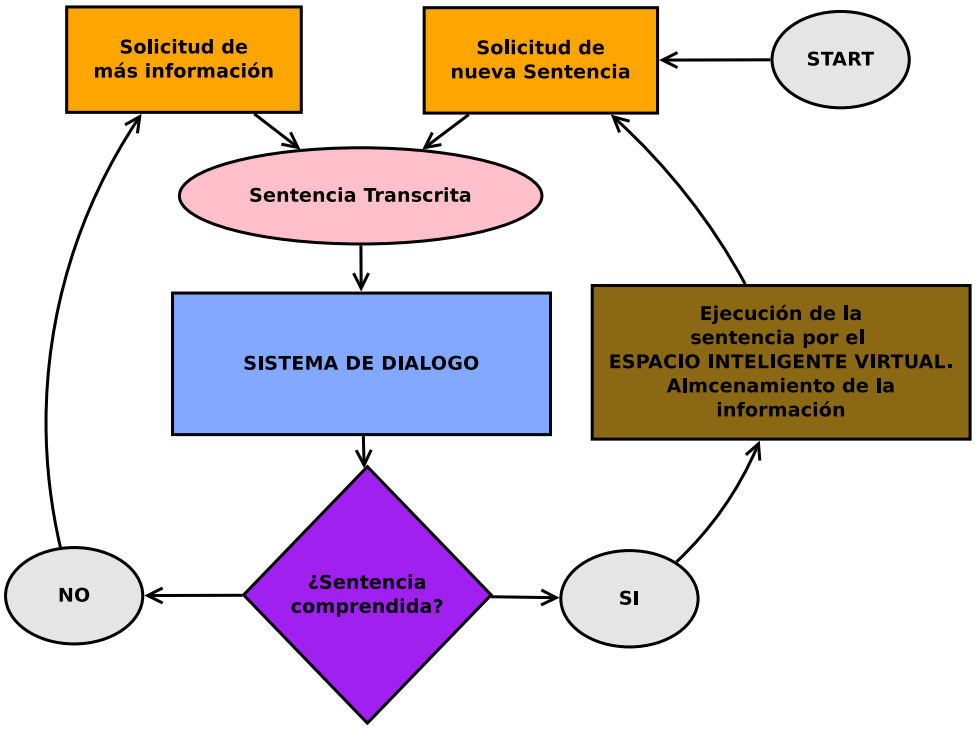


Figura 3.14: Algoritmo de ejecución del sistema de diálogo simplificado.

resuelta en un futuro próximo. A modo de ejemplo, las sentencias `ABRE` es un ejemplo de omisión de un objeto que el sistema de diálogo podría resolver fácilmente preguntándole al usuario por el objeto que desea abrir. De la misma forma, la sentencia `ABRE LA LAMPARA` es un ejemplo de sentencia en los que la acción y el objeto no concuerdan semánticamente pero las palabras de la sentencia sí que pertenecen al vocabulario de la aplicación. Para resolverse se pueden seguir varias estrategias. Una de ellas, podría ser preguntar al usuario por el objeto que se desea abrir y por la acción que se desea ejecutar sobre el objeto lámpara.

Las sentencias erróneas con información parcial se pueden resolver en sucesivas iteraciones del algoritmo como hemos visto. Una de las técnicas más conocidas para llevar a cabo esta tarea consiste en la implementación del algoritmo *blackboard* o algoritmo de la pizarra. A continuación se explicará en que consiste el algoritmo *blackboard*.

3.6.1 El algoritmo *blackboard*

El algoritmo *blackboard* se puede explicar de forma intuitiva acudiendo al juego del bingo. En el juego del bingo, los jugadores tienen que ir marcando en su cartón los números que las bolas van indicando. En cada iteración, una bola se extrae del bombo. Cuando un jugador completa la línea por primera vez en el juego canta la línea, y el primer jugador que complete todos los números de su cartón canta bingo y gana el juego. El algoritmo *blackboard* funciona de forma similar. Cuando nos llega sentencia errónea, el algoritmo detecta qué campos tienen sentido y los almacena en una estructura de datos conocida como *blackboard* o pizarra debido a su analogía con las pizarras de clase. Después, en la siguiente iteración, el algoritmo pregunta al usuario por la sentencia errónea, de tal modo que con la nueva información aportada junto con la almacenada en la *blackboard* puede conformar una sentencia comprensible por el sistema y en consecuencia ejecutable.

En el diagrama de flujo de la figura 3.15 se representa el algoritmo de *blackboard* completo, así como un ejemplo que esclarezca la explicación.

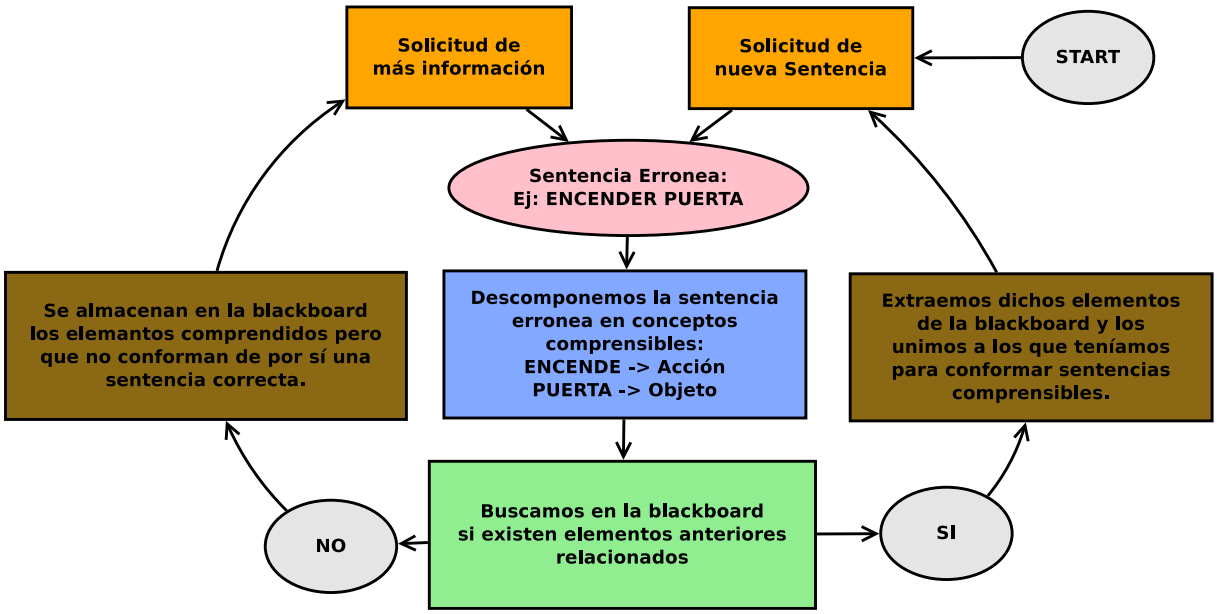


Figura 3.15: Algoritmo *blackboard*.

Existen múltiples criterios y maneras de almacenar la información en la *blackboard*. En nuestro caso particular, la información se almacena clasificándola por objetos, acciones o modificadores incompletos. A modo de ejemplo, si en la sentencia errónea falta un objeto asociado a la acción `ABRIR`, se almacena la acción en la casilla "acción incompleta" a la espera de en las sucesivas iteraciones llegue el objeto asociado con dicha acción. La figura 3.16 muestra la estructura de datos de una *blackboard* clásica.

El esquema clásico del algoritmo *blackboard* no contempla más que un único almacenamiento en cada campo de la estructura de datos, véase la figura 3.16. Es decir, particularizando para nuestro caso, en la *blackboard* solamente se podrá almacenar una acción, un objeto y un modificador sin modificar. Si la *blackboard* contiene algún elemento no resuelto, como por ejemplo la acción `ABRIR`, y en la siguiente iteración el usuario no pronuncia el objeto correspondiente para resolverlo, sino que emite la acción *CERRAR*, este elemento se sobrescribe perdiéndose la primera palabra emitida. Este comportamiento puede ser beneficioso, sobre todo si el usuario se ha equivocado al emitir ese comando y simplemente en la siguiente iteración trata de borrarlo. Sin embargo, si se permiten sentencias complejas en las que pueda aparecer más de una acción en la misma sentencia estaremos perdiendo información útil. Es el caso de la sentencia errónea `ABRIR ENCENDER`, donde solamente `ENCENDER` se almacenaría en la `blackboard`. Nos gustaría que el sistema de diálogo respondiese ante este tipo de sentencias contestando al usuario una sentencia del tipo `¿Qué objeto desea abrir?` y `¿Qué objeto desea encender?`.

Figura 3.16: Estructura blacboard clásica.

En nuestro caso particular, las sentencias complejas están permitidas, véase la sección 3.3 en la página 3.3. Por esta razón, es conveniente alejarnos del modelo clásico del algoritmo *blackboard* y modificarlo

ligeramente para que no se produzca dicho efecto.

3.6.2 El algoritmo *blackboard* modificado: La pila de pizarras

La primera técnica que se nos ocurre para mejorar el algoritmo *blackboard* es la de hacer que la estructura de datos de la *blackboard* sea una lista de datos. De esta forma, si en una determinada iteración del algoritmo se produce una sentencia errónea parcialmente comprensible esta se almacenará en la primera posición de la lista. Si después, en la siguiente o las siguientes iteraciones, se recibiera otra sentencia errónea, se almacenaría en las siguientes posición de la lista, de ese modo se evita la pérdida de la información anterior. En la imagen de la figura 3.17 se muestra este tipo de estructura.

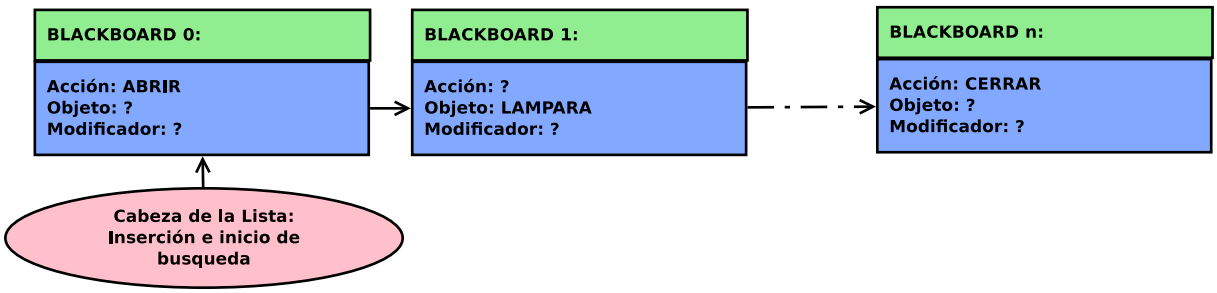


Figura 3.17: Estructura *blackboard* modificada.

Como se observa en el gráfico de la figura 3.17, tanto la inserción de la información parcial comprendida como la búsqueda de información en la *blackboard* comienza por la cabeza de la lista. La razón de esta forma de recorrer la lista es la aplicación de la política *LAST IN FIRST OUT (LIFO)*. La política LIFO es la política más acertada de almacenaje y recorrido de datos. La razón es sencilla de entender. Cuando se recibe una sentencia errónea y ésta es detectada, el sistema de diálogo empieza a buscar por la información más recientemente recibida. Ocurre que normalmente, cuando se produce una sentencia errónea y se informa al usuario, el propio usuario normalmente resolverá el error en la siguiente iteración y no esperará varias iteraciones para resolverla. Este es el comportamiento usual de un usuario y por esta razón, la política más acertada que deberemos utilizar es la LIFO

La estructura de la figura 3.17 tiene alguna desventaja que debemos solventar. El problema principal radica en que la lista puede llenarse indefinidamente al no estar limitada su longitud. Esto es un problema porque el algoritmo del sistema de diálogo se esta ejecutando de manera continuada. Al cabo del tiempo, pueden quedar residuos de sentencias erróneas que el usuario no ha querido resolver o resolvió introduciendo una sentencia correcta. El caso que se presenta a continuación muestra un claro de este hecho: Supongamos que el usuario ha emitido la sentencia `CIERRA`. El sistema de diálogo, al detectar la palabra errónea, informa al usuario del error y el usuario reacciona respondiendo `CIERRA LA PUERTA`. La ultima sentencia es una sentencia comprensible por el sistema y resuelve el error sin acudir a la `blackboard`. La primera acción cerrar es un buen candidato para que la acción se quede almacenada para siempre en la *blackboard* tal y como esta diseñada la estructura de la figura 3.17.

Por esta razón, se ha diseñado un mecanismo que elimine los elementos de la *blackboard* cuando estos permanecen en ella demasiado tiempo. Se ha implantado un mecanismo de envejecimiento por el cual, ningún elemento puede estar más de un determinado número de iteraciones en la lista. Con cada iteración, se incrementa la edad de cada uno de los elementos en la *blackboard*. Al llegar a un límite, para nuestro caso particular ese límite esta situado en 5 unidades, el elemento es sacado de la lista solucionando la posible acumulación de elementos en la lista. El límite escogido ha sido seleccionado de forma experimental. Se supone que en menos de 5 iteraciones el usuario habrá solucionado todos los posibles errores pertinentes. La imagen de la figura 3.18 representa este tipo de estructura alternativa con envejecimiento.

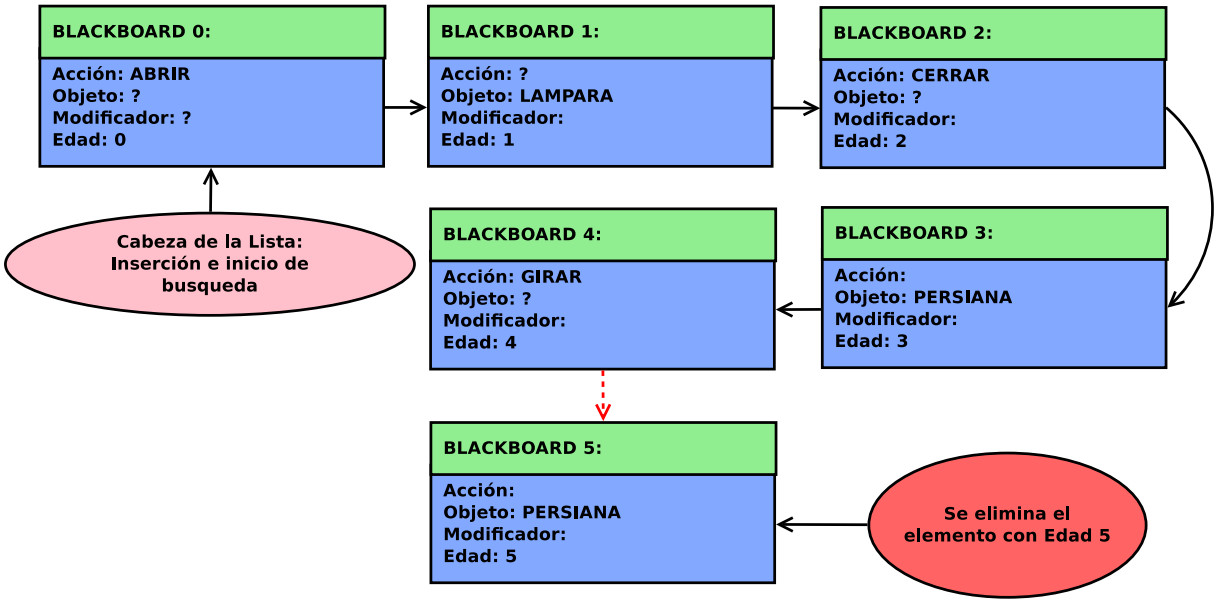


Figura 3.18: Estructura *blackboard* modificada con envejecimiento.

Dotar de memoria al sistema de diálogo añade flexivilidad y versatilidad al sistema produciendo diálogos más naturales, más humanos. En realizar, lo que estamos haciendo a la hora de introducir esta lista es crearnos un mapa temporal que almacena información del pasado, al igual que el los seres humanos, al dialogar, recuerdan lo que a dicho su interlocutor con anterioridad.

Una vez explicado el algoritmo *blackboard* y las modificaciones realizadas para resolver los problemas planteados, en la siguiente sección nos vamos a centrar en describir el algoritmo completo del sistema de diálogo.

3.6.3 El algoritmo complejo del sistema de diálogo

Una vez explicado el algoritmo *blackboard* (descrito en la sección 3.6.3) y mostrado el diagrama de flujo simplificado del algoritmo de ejecución del sistema de diálogo (mostrado en la figura 3.14), vamos a centrarnos en el algoritmo completo. En la figura 3.19 puede verse el diagrama lógico de ejecución del algoritmo.

El esquema a priori puede parecer muy complicado pero según se vaya explicando todos los pasos que se llevan a cabo se irá esclareciendo. En el diagrama de flujo de ejecución, los procesos que lleva a cabo el algoritmo están numerados. Nosotros seguiremos cada uno de estos puntos a la hora de explicar el algoritmo:

1. El primer bloque del algoritmo es el comienzo de la ejecución. En este punto, el algoritmo espera recibir una sentencia transcrita de los reconocedores de voz y de gestos como mostrábamos en la figura 3.12. Una vez recibida la sentencia por parte de los reconocedores, comienza el algoritmo.

2. El segundo bloque de ejecución es un preprocesado de la información recibida. La sentencia recibida por parte de los reconocedores puede llegar con un formato distinto al que espera recibir el sistema de diálogo. Para nuestra aplicación, la sentencia recibida se procesa con letras mayúsculas y sin acentos. Una vez normalizada la sentencia con nuestro propio formato, el siguiente paso de este bloque de ejecución consiste en realizar un conteo de las palabras que contiene. Es aquí donde surge nuestro primer escollo: las palabras que queremos contar son aquellas que contienen significado

Figura 3.19: Algoritmo de ejecución completo del sistema de diálogo.

gramatical y semántico. El reconocedor de voz en concreto, es capaz de transcribir sentencias orales en inglés y en castellano. Con el castellano no tenemos ningún problema, pero en ingles, existen conjuntos de palabras como `TURN ON`, `SWITCH OFF`...que para nuestra aplicación particular se consideran una sola puesto que su significado se crea conjuntamente. Este bloque por lo tanto debe contar no simplemente las palabras por que estén separadas por espacio, sino que tendrá en cuenta la consideración anterior. Una vez acabado acabada la ejecución del conteo, este bloque proporciona tanto la sentencia transcrita normalizada como el número de palabras detectadas e inicializa la cabecera de una lista de tipo `TUnderstoodSentence` como la de la figura 3.11 de la sección 3.5.2 , donde se almacenará la información comprendida de dicha sentencia.

3. El tercer bloque recibe la sentencia ya normalizada y la recorre buscando posibles acciones que encontremos en la sentencia. El algoritmo de búsqueda recorre la estructura de datos (véase la figura 3.10) que almacena la gramática y que vimos en la sección 3.5 en la página 28. Por cada una de las acciones encontradas en la sentencia de entrada (recordemos que la sentencias de entrada podían ser complejas y por lo tanto tender más de una acción), este bloque funcional genera un elemento de tipo `TUnderstoodAction` (mostrado en la figura 3.11) que se rellena con la información almacenada en el objeto de la clase `cAction` (mostrado en la figura 3.10) correspondiente y se enlaza en la lista enlazada `TUnderstoodSentence` inicializada en el punto anterior. A modo de ejemplo, si la sentencia de entrada es `ABRE LA PUERTE Y ENCIENDE LA RADIO`, este bloque generará un elemento de tipo `TUnderstoodAction` que rellenará con la acción `ABRE` y otro objeto con la acción `ENCIENDE`. Estos elementos se agregarán dentro de la cabecera de tipo `TUnderstoodSentence` y se podrá el indicador de número de acciones encontradas a 2. En el código del algoritmo que se adjunta, este bloque funcional lo lleva a cabo la función cuyo prototipo se muestra en la figura 3.20.

```
1       int __findActionsInSentence(string action, vector<c_Action> &v_acciones,
            TUnderstoodSentence* p_undSent);
2       /* action: Sentencia donde se buscará acciones
3          v_acciones: Vector de acciones c_Action donde se almacena la información de la
               gramática
4          p_undSent: Puntero donde se creará la cabecerá de la lista de acciones que se
               encontrarán.
5       */
```

Figura 3.20: Función de búsqueda de acciones

Si la sentencia no contiene ninguna acción, entonces, la cabecera de la lista `TUnderstoodSentence` tendrá un puntero apuntando a `NULL` y el contador de acciones asociado valdrá 0.

Después de ejecutar este punto, sabremos por tanto si la sentencia contiene o no acciones que ejecutar. El bloque ofrece tanto el número de acciones en la sentencia como información sobre las mismas. Toda esta información se almacena en una lista de tipo `TUnderstooodSentence`.

4. Como vemos en la figura 3.19, este bloque se ejecuta siempre que vengamos del bloque funcional 3 y el número de acciones comprendidas sea al menos 1. Al llegar a este punto del algoritmo, lo primero que se hace es analizar los objetos asociados a cada una de las acciones que nos hemos encontrado en el punto anterior y rellenamos la lista en consecuencia. Para el caso de la sentencia `ABRE LA PUERTA Y ENCIENDE LA RADIO`, las acciones que se encontraron son `ABRE` y `ENCIENDE`. Ahora, el objeto `PUERTA` se debe asociar a la acción `ABRE` mientras que el objeto `RADIO` debe asociarse a `ENCIENDE`. El módulo rellena los campos correspondientes de los objetos de tipo *TUnderstoodAction* que guarda la lista de objetos. Este bloque funcional además rellenar los objetos asociados, devuelve

el número de objetos que ha asociado a las acciones. La función que realiza esta función es la que se muestra en la figura 3.22

```
1       int __analizeObject(string obj, vector<c_Action> &v_acciones, TUnderstoodSentence
            * p_undSent);
2       /* object: Sentencia donde se buscará acciones.
3          v_acciones: Vector de acciones c_Action donde se almacena la información de la
               gramática.
4          p_undSent: Puntero donde se añadirá la información de los objetos.
5       */
```

Figura 3.21: Función de búsqueda de de objetos y asociación.

5. Después de buscar los objetos, el algoritmo en este punto realiza las misma operaciones en busca de modificadores. El bloque funcional lo agrega a la lista y devuelve el número de modificadores asociados.

```
1       int __analizeModifier(string mod, vector<c_Action> &v_acciones,
            TUnderstoodSentence* p_undSent);
2       /* mod: Sentencia donde se buscará acciones.
3          v_acciones: Vector de acciones c_Action donde se almacena la información de la
               gramática.
4          p_undSent: Puntero donde se añadirá la información de los objetos.
5       */
```

Figura 3.22: Función de búsqueda de modificadores y asociación.

6. El bloque funcional 6 solamente se ejecuta si existe algún objeto o algún modificador que no haya sido asociado a ninguna acción. Puede ocurrir, es el caso de sentencias como `ABRE LA PUERTA Y ROBOT`. Hasta este punto se hubiese procesado la acción `ABRE` y el objeto `PUERTA` asociado a ella pero el objeto `ROBOT` no estaría asociado a ninguna acción y se perdería. Para detectar esta situación, tan solo tenemos que restar al contador de palabras detectadas, el contador de acciones y el contador de objetos y modificadores asociados. Si el resultado fuese positivo, tenemos que averiguar que objeto o que modificador está sin asociar e insertalo en la `blackboard` para su posterior actualización. La Las funciones que realizan esta tarea son las que se muestran en la figura 3.23.

```
1       int __findObjectsInSentence(string sentence, vector<c_Action> &v_acciones,
            TUnderstoodSentence* p_possibleActionsBindObject);
2       /* sentencia: Sentencia donde se buscará objetos.
3          v_acciones: Vector de acciones c_Action donde se almacena la información de la
               gramática.
4          p_undSent: Puntero donde se añadirá la información de los objetos.
5       */
6       int __findModifiersInSentence(string sentence, vector<c_Action> &v_acciones,
            TUnderstoodSentence* pModifierSentence);
7       /* sentencia: Sentencia donde se buscará modificador.
8          v_acciones: Vector de acciones c_Action donde se almacena la información de la
               gramática.
9          p_undSent: Puntero donde se añadirá la información de los modificadores.
10      */
```

Figura 3.23: Funciones de búsqueda de objetos y modificadores y creación de una estructura `TUnderstoodAction`.

La estructura de tipo `TUnderstoodAction` rellena los campos relacionados con la acción con -1 o con una cadena vacía. De esta forma al recorrer, la blackboard el algortimo en busca de una acción, sabrá identificar que el elemento es un objeto de forma fácil.

7. El algoritmo ejecuta el bloque 7 si viene del bloque 5 o del bloque 6 y no ha encontrado objetos asociados con la acción. En este punto, el algortimo busca posibles objetos en la `blackboard`.

8. En este punto, el algoritmo ha encontrado un objeto en la `blackboard` que encaja con la acción que se estaba analizando. El algortimo extrae el elemento de la `blackboard` para formar una sentencia comprensible. También, almacena la frase correcta en la zona de memoria de sentencias correctas para su posterior almacenamiento.

9. Si no se encuentra ningún objeto en la `blackboard` que encaje con la acción que se está procesando, el algoritmo almacena la acción en la blackboard para una posible resolución futura. A modo de ejemplo, si el usuario enuncia la palabra `ABRE` y no encuentra un objeto en la `blackboard`, el algoritmo la introduce en la misma.

10. Ahora volvemos atrás en el grafo de la figura 3.19, concretamente al punto 10 que se ejecuta solamente cuando no se detecta ninguna acción en la sentencia. En este punto entraremos con sentencias del tipo `VENTANA` o `LAMPARA Y LUZ`. Este bloque funcional realiza una búsqueda de todos los objetos de la frase aunque no esten asociados a ninguna acción. Genera una estructura de tipo `TUnderstoodAction` por cada objeto encontrado sin asociar y rellena tanto el identificador de acción asociada a -1 como el nombre del identificador a $NULL$. La función que se ejecuta en este punto es la misma que la del punto 6, véase la figura 3.23. Como resultado de este punto tendremos una lista de tipo `TUnderstoodSentence` en la que se enlazarán los objetos encontrados, así como un contador de objetos sin asociar de elementos

11. En este punto se desarrolla la misma labor que en el punto anterior salvo que en este caso se buscan los modificadores. La función que ejecuta el algoritmo se mostró en la figura 3.23.

12. Si se encuentra algun objeto, el siguiente paso consistirá en recorrer la `blackboard` con el objetivo de encontrar la acción que encaje con el objeto sin asociar. Esta es la labor que se ejecuta en dicho bloque funcional.

13. Suponiendo que no se encuentre ningun objeto sin asociar, la sentencia no será validad dado el esquema de la gramática. Ejemplos de este tipo de sentencia será `ARRANCA EL COCHE`. Ni la acción `ARRANCA`, ni el objeto `COCHE` pertencen al vocabulario. Por lo tanto, este bloque es un bloque funcional de aviso de esta situación.

14. Este bloque se ejecuta cuando venimos del punto 12 y hemos encontrado en la `blackboard` una acción que sí que encaja con el objeto que aparecía en la sentencia. Ejemplo de sentencia que provoca la entrada a este bloque funcional es la sentencia `VENTANA`, siempre que con anterioridad se halla emitido una sentencia como `ABRIR` que haga que tengamos en la `blackboard`. Entre las labores que el algoritmo debe realizar en este punto están la de sacar la acción de la `blackboard` y la de conformar la sentencia correcta con la información actual.

15. Este bloque es la antítesis del bloque anterior. En este caso, al buscar en la `blackboard` la acción relacionada con el objeto recibido, el algoritmo no encuentra ninguna. En este bloque, por lo tanto, se inician las tareas para insertar el objeto en la `blackboard`. Sentencias que ejecutan este módulo son por ejemplo `PUERTA` sin previamente haber dicho ningua acción relacionada con ella.

16. Este es el punto final al que se llega despues de ejecutar el algoritmo, vengas de la rama que vengas. En este punto se realizan las siguientes tareas:

    - En primer lugar se incrementa el contador de edad de todos los elementos de la *blackboard* y se eliminan aquellos elementos cuya edad sobrepasen las 5 iteraciones. Si miramos el gráfico

de la figura 3.19, vemos como se ha completado una iteración del algoritmo. Es justamente en este momento donde se debe llevar a cabo dicha gestión.

- Se procesan los errores que se hallan producido durante el proceso, los cuales hacen que se aleje de la sentencia correcta que debería haberse recibido. En este momento, se debe de comprobar porqué motivo se ha descartado la sentencia de entrada como errónea.
- Se informa de los errores que existen en la sentencia recibida y se pregunta al usuario información para que sea resulta. Para ello, se escribe por pantalla una serie de sentencias interrogativas en función del nivel de experiencia del usuario del sistema.

17. El punto 17 se ha resaltado al final porque es el caso normal de una sentencia recibida correctamente. Es el caso de sentencias simples como `ABRE LA PUERTA` o complejas como `ABRE LA PUERTA Y ENCIENDE LA VENTANA`, las cuales encajan a la perfección con la gramática definida.

Una vez se ha ejecutado el algoritmo completo se llama al generador de respuestas, cuyas características se explicarán en la sección 3.8 para informar al usuario de los resultados de cada iteración. A continuación, se va a explicar una técnica de confirmación muy útil que hemos diseñado en nuestra aplicación: la técnica de cancelación.

3.7 Técnica de cancelación

Antes de explicar en qué consiste la técnica de cancelación debemos explicar en que consisten las técnicas de confirmación. Una técnica de confirmación es un mecanismo que permite al sistema de diálogo cerciorarse de que aquella información que ha comprendido es la misma que la que el usuario quería comunicarle. El sistema de diálogo no es un sistema perfecto libre de errores. Para empezar, acarrea todos los errores que pueda cometer los reconocedores de los que se sirve. Estos ya de por sí trabajan con una probabilidad de error no nula aunque sí muy baja. Por otro lado, si el sistema de diálogo trata de resolver sentencias incompletas (erróneas desde el punto de vista de sentencia estándar), puede que no acabe de resolver la sentencia aun incluyendo funcionalidades que lo permite. Por esta razón, se incluyen métodos de confirmación que permiten validar si la sentencia que ha interpretado el sistema de diálogo es la que se quería comunicar. Un ejemplo de método de confirmación sería el siguiente: El sistema de diálogo recibe la sentencia `ABRE LA VENTANA` y pregunta si la sentencia comprendida `ABRE LA VENTANA` es la correcta. El usuario responderá al sistema confirmando la comprensión de la sentencia.

La técnica de cancelación es un subtipo de método de confirmación pero en negativo, es decir, se entiende que las sentencias que comprende son correctas hasta que el usuario diga lo contrario. Este método es ventajoso en escenarios donde en la mayor parte de los casos los reconocedores reconocen correctamente las sentencias, o donde el sistema de diálogo llega a resolver las sentencias erróneas en la mayoría de los casos. El razonamiento que se ha utilizado es el siguiente: Si en la mayor parte de los casos, las sentencias comprendidas son correctas, entonces, se hace tedioso que tengamos que estar preguntándole al usuario si la misma es correcta o no. El sistema puede suponer que el la sentencia es correcta y continuar con la ejecución del programa de tal modo que si en la siguiente ejecución del algoritmo recibe una señal de cancelación, sabrá que no ha sido validada. Si el usuario simplemente sigue dialogando, el sistema da por válida la anterior sentencia y continúa con la ejecución de su algoritmo. Esta forma sencilla facilita el diálogo con el usuario, haciéndolo más cómo al evitar interrupciones innecesarias.

Para la implementación de esta técnica, simplemente se ha introducido una palabra dentro del vocabulario del propio sistema de diálogo: `CANCELAR` o su homólogo en inglés `CANCEL`. La palabra `CANCEL` no es más que una acción como otras de las ya enunciadas (`OPEN`, `CLOSE`...), y por lo tanto, para seguir

la estructura de la gramática que vimos en la sección 3.3 se debe de acompañar de un objeto asociado como por ejemplo `ACTION`, `SENTENCE`... Cuando el sistema de diálogo, una vez se ejecute el algoritmo mostrado en la figura 3.19, haya comprendido la sentencia `CANCELAR`, entonces podrá hacer 2 cosas en función de la situación en la que se encuentre:

1. Si la sentencia recibida en la iteración anterior fue una sentencia correcta, es decir, gramaticalmente es correcta, entonces, el gestor del diálogo procederá a cancelar la ejecución de la sentencia anterior. A modo de ejemplo, si una iteración el usuario dijo la frase: `ABRE LA PUERTA` y se comprendió la frase `ABRE LA VENTANA`; si en la siguiente iteración del algoritmo, el usuario ha dicho `CANCELAR SENTENCIA` y se ha comprendido correctamente, entonces, la frase anterior quedará invalidada.

2. Si la sentencia recibida y comprendida con anterioridad no era correcta, como por ejemplo, `ABRE`, si el usuario emite después el comando `CANCELAR SENTENCIA`, entonces borrar todos los elementos almacenados en la *blackboard*. De este modo, el sistema deja de intentar resolver la frase puesto que se entiende que no ha acertado desde el principio y el usuario quiere volver a empezar.

La técnica de cancelación mejora de forma decisiva el sistema de diálogo. Se mostrarán algunos ejemplos en el apartado 4.6 de resultados.

3.8 El módulo generador de respuestas

Una vez ejecutado el algoritmo y comprendida la sentencia recibida, es el momento de informar al usuario de los resultados de dicha iteración del algoritmo. El módulo generador de respuestas se encarga de generar las sentencias que, o bien se mostrarán simplemente por pantalla para informar al usuario, o bien servirán como entrada al sintetizador correspondiente para que la reproduzca.

Con cada iteración del algoritmo de diálogo, el propio algoritmo al finalizar llama al generador de respuestas. Independientemente de si la sentencia recibida es correcta, parcialmente correcta, o incorrecta, el generador de respuestas es llamado y comienza ala ejecución de su código. El generador de respuestas funciona de la siguiente manera:

- La primera labor que desarrolla el generador de respuestas es recorrer la lista de tipo *TUnderstood-Sentence* que almacena la información comprendida de la sentencia actual. Busca si la sentencia es correcta o si por el contrario es errónea. Si esta fuese errónea, además busca la razón por la cual es errónea, por ejemplo porque falte el objeto asociado a la acción, y asocia en consecuencia un código de error a esa frase. A modo de ejemplo, en una iteración cualquiera se comprendió el comando `ABRE LA PUERTA`. El generador de respuestas le dio código 0 al ser correcta la sentencia comprendida.

- El error es interpretado por el generador de respuestas emitiendo una sentencia en función de dos parámetros distintos: El modo y el aleatorizador. A modo de ejemplo, teniendo presente que el sistema de diálogo esta configurado en modo cadete y que el aleatorizador de datos ha dado el valor 1, la sentencia que emite es la siguiente: `He comprendido la orden: ABRE LA PUERTA`.

- Finalmente, una vez se ha emitido la sentencia el generador de respuestas pasa el testigo a la espera de la siguiente iteración.

En las siguientes secciones se va a explicar en que consiste el modo de uso y el aleatorizador

3.8.1 El modo de uso

El modo de uso es una técnica que permite generar las sentencias respuesta en función del tipo de usuario o del rol que desempeñe. Nosotros hemos diseñado dos tipos de roles:

- El modo cadete presupone un tipo de usuario con poco conocimiento sobre el sistema de diálogo y por tanto se le debe de informar con todo detalle de los posibles errores que haya cometido a la hora de emitir sentencias. Un ejemplo de este tipo de respuesta es la siguiente: `Usted quiere ejecutar la acción: ABRIR pero no se ha entendido un objeto que este relacionado con ella. Por favor, exprese el objeto sobre el que recaerá la acción: ABRIR.`
- El modo veterano presupone un usuario experto con amplio conocimiento de las razones por las cuales el sistema puede fallar y en consecuencia le parece redundante y pesado que le estén recordando de forma exaustiva todo el rato los fallos que comete. Una sentencia respuesta para este tipo de usuario sería `Acción: ABRIR sin objeto.`

El hecho de que distingamos entre diferentes tipos de usuarios mejora la versatilidad del sistema y la adaptación a los diferentes tipos de usuarios que el sistema se pueda encontrar.

3.8.2 El aleatorizador de respuestas

El aleatorizador de respuestas no es más que una máquina que genera números aleatorios de tal modo que cuando el generador de respuestas debe emitir una sentencia, se elige una de entre un conjunto prototipo. En general, las personas no repiten siempre el mismo patrón de respuesta ante las preguntas que se le formulan. Existen variaciones que permiten evitar esta repetitividad tan característica de los diálogos robóticos.

Para su implementación, lo que se ha hecho ha sido escoger un conjunto de sentencias prototipo que se corresponde con cada uno de los casos a resolver. De tal modo que en cada iteración, aleatoriamente, se escoja una sentencia del conjunto. A modo de ejemplo, a continuación se muestra las posibles sentencias que se mostrarían en el caso de que la sentencia comprendida fuese correcta:

- Entendido! Usted ha ordenado el comando: “SENTENCIA COMPRENDIDA”.
- He comprendido la orden: “SENTENCIA COMPRENDIDA”.
- Perfecto! La orden recibida es: “SENTENCIA COMPRENDIDA”.
- Eureka! Entendido: “SENTENCIA COMPRENDIDA”.

Existe un conjunto similar para cada caso erróneo detectado, y para cada uno de los roles de los que hablamos en la sección 3.8.1. El uso del aleatorizador proporciona al sistema un diálogo más natural. Las repeticiones de estructuras suelen ser tediosas tanto en el diálogo entre humanos como el que pordamos entablar con máquinas.

3.9 Conclusiones

En este capítulo de desarrollo hemos visto cuales son los pasos principales que se deben de dar para construir un sistema de diálogo. Las primeras tareas pasan por la definición de un vocabulario y una

gramática diseñada para una aplicación en concreto. Después, vimos que la siguiente tarea consiste en difinir cuales son los errores más frecuentes con los que el sistema se debe encontrar para solucionarlos. Mas tarde hablamos de las ventajas e inconvenientes de las diferentes estructuras de programación que debemos utilizar para recorrer el grafo de la gramática para acercarnos al caso óptimo. En última instancia se vio el diagrama de flujo del sistema de diálogo diseñado. Este esquema resuelve tanto las sentencias correctas como los principales errores que el usuario o el reconocedor pueden cometer.

Capítulo 4

Resultados

La conversación más agradable es aquella de la que no se recuerda nada con precisión, pero deja una impresión general agradable.

Ben Jonson

4.1 Introducción

En este capítulo se introducirán los resultados más relevantes del trabajo. Haremos una exposición concisa tanto de los errores que nuestro sistema de diálogo es capaz de resolver como de los que no puede evitar. Además, mostraremos el funcionamiento del algoritmo *blackboard* (véase la figura 3.6.3), la técnica de cancelación de la que ya hablamos (véase la figura 3.7), y el funcionamiento del generador de respuestas (véase la figura 3.8). Todos estos módulos fueron explicados en el capítulo 3 de desarrollo .

El capítulo está estructurado en las siguientes secciones:

1. En la primera sección, hablamos sobre la metodología y las pruebas que se han realizado para desarrollar la evaluación de nuestro sistema de diálogo.
2. En la segunda sección, expondremos algunos ejemplos de como se comporta el diálogo al recibir sentencias correctas. En ella veremos como el sistema de diálogo se comporta como se esperaba ante este tipo de sentencias.
3. En el tercer apartado veremos como el sistema de diálogo es capaz de detectar algunos errores que se pueden producir en la comunicación y como el sistema es capaz de resolverlos.
4. En la cuarta sección daremos algunos ejemplos de funcionamiento de la *blackboard* y veremos el mecanismo de envejecimiento de sus elementos.
5. La quinta sección mostrará diferentes ejemplos del funcionamiento de la técnica de cancelación.
6. La ultima sección mostrará los resultado obtenidos por parte del generador de respuestas.

4.2 Metodología

Antes de pasar a ver los resultado obtenidos tenemos que conocer realmente el sistema de diálogo que estamos evaluando, para después aplicar en consecuencia el método óptimo de evaluación.

En primer lugar debemos saber que el sistema de diálogo que hemos diseñado y desarrollado es un sistema determinista, es decir, su búsqueda recorre la sentencia tratando de encontrar estructuras gramaticales por fuerza bruta. Como vimos en el capítulo 2, los sistemas de diálogo pueden ser de tipo estocástico, es decir, basarse en un modelo de probabilidades para encontrar la sentencia más probable. Para la evaluación de estos últimos se calculan probabilidades de acierto o tasas de fallo, sin embargo, un sistema determinista nunca falla o al menos tal no falla para aquello para lo que fueron diseñados. Por esta razón, para evaluar un sistema determinista basta con realizar una serie de pruebas y comprobar el correcto funcionamiento del mismo.

En las siguientes secciones se mostrarán precisamente las pruebas que hemos realizado a este sistema y como se comporta a la mismas.

4.3 Pruebas de sentencias correcta

En este apartado se quiere dar una visión global del comportamiento del sistema de diálogo. Por ello, se han llevado a cabo una serie de pruebas en las que el sistema de diálogo debe responder a un usuario que siempre emite sentencias correctas, es decir, un usuario ideal. Este usuario no se aleja en exceso de un usuario real puesto que la probabilidad de que el usuario emita una sentencia correcta es elevada. No nos encontramos en un entorno hostil.

En el primer apartado se muestra el comportamiento ante sentencias sencillas y poco a poco se van complicando las pruebas.

4.3.1 Sentencias simples

En esta sección mostramos como el sistema es capaz de comprender sentencias simples bien construidas. Recordemos que las sentencias simples son aquellas que poseen una única acción. En la figura 4.1, el sistema pregunta al usuario qué desea. El usuario inserta la sentencia *OPEN THE DOOR* y el sistema comprende tanto la acción *OPEN* como el objeto *DOOR* asociado a ella.

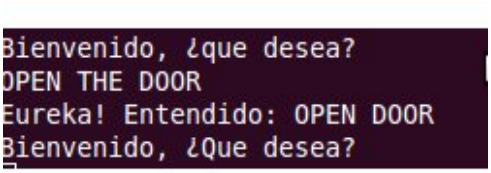


Figura 4.1: Ejemplo de sentencia simple.

El modificador *THE* es obviado puesto que no aporta información alguna al significado de la frase. La sentencia que se muestra se realiza a partir de la lista que contiene todos los elementos comprendidos en esa misma iteración.

Otro ejemplo de sentencia simple un poco más elaborado es el que se muestra en la figura 4.2. En este ejemplo, tenemos otra frase como la anterior pero esta vez a la acción le acompaña más de un objeto. En este caso, a la acción `OPEN` le acompañan los objetos *DOOR* y *WINDOWS*. El sistema de diálogo comprende tanto la acción como los objetos asociados con ella, omitiendo los modificadores carentes de significado.

Vamos a ver un caso más complicado, una sentencia compleja con dos o más acciones.

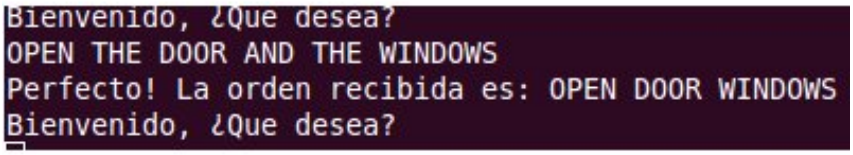


Figura 4.2: Ejemplo de sentencia simple 2.

4.3.2 Sentencias complejas

Las sentencias complejas recordemos que eran sentencias en las que aparecía más de una acción. En la figura 4.3 se muestra una sentencia de este tipo.

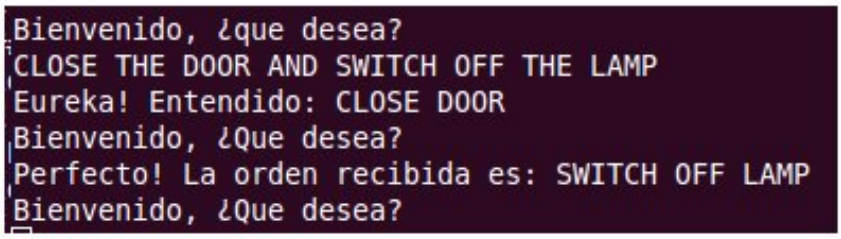


Figura 4.3: Ejemplo de sentencia compleja.

La sentencia compleja es desglosada en dos sentencias simples que son comprendidas por el sistema de diálogo. Como vemos, el sistema de diálogo parece comprender todas las sentencias que el usuario está emitiendo, al menos, las que se ajustan al modelo estándar. La figura 4.4 muestra una sentencia más elaborada. En ella se observan tres sentencias simples, una de las cuales posee más de un objeto asociado.

Como puede observarse, el sistema de reconocimiento consigue sin dificultad comprender la frase introducida.

A continuación, vamos a ver como se comporta el sistema de diálogo en los caso en la sentencia de entrada es errónea.

4.4 Pruebas de sentencias erróneas

En esta sección vamos a ver como el sistema de diálogo es capaz de detectar los errores para los cuales había sido diseñado y como es capaz de lanzar preguntas al usuario para tratar de corregirlos. Comenzaremos viendo los errores en sentencias simples para después pasar a los errores en sentencias más complejas.

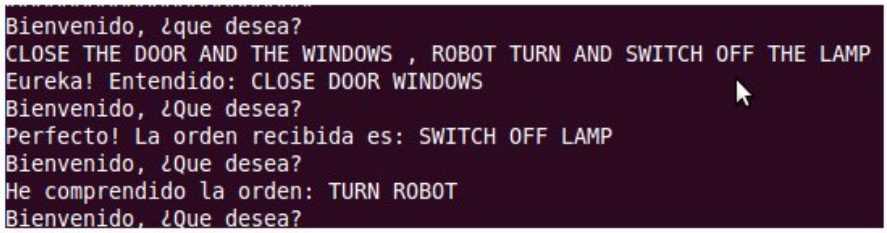


Figura 4.4: Ejemplo de sentencia compleja.

4.4.1 Errores en sentencias simples

Las sentencias simples, si recordamos el capítulo 3, son sentencias con una única acción y uno a varios objetos asociados a ellas. Los posibles errores que podemos cometer en una acción simple es que, o bien no hayamos introducido el objeto asociado a la acción introducida, o bien que se halla introducido un objeto sin asociarse a ninguna acción. También puede ocurrir el caso en el que el usuario haya introducido una acción y un objeto que no estén relacionados entre sí. Esta clase de error se verá más adelante cuando se traten los errores en sentencias complejas.

Veremos con ejemplo como el sistema de diálogo es capaz de detectar los errores en sentencias simples y corregirlos en consecuencia.

4.4.1.1 Omisión del objeto

Si recordamos, la sentencia ideal debía llevar una acción y un objeto asociado. Si el usuario emite una sentencia en la que solo aparece una acción como por ejemplo la que se muestra en la figura 4.5, el sistema es capaz de detectarlo y actuar en consecuencia para resolver esta situación. En el caso de la figura 4.5, el usuario ha emitido la sentencia `ABRE` y el sistema ha detectado esta situación errónea respondiendo al usuario por el objeto que ha omitido. El sistema emite la respuesta: `Es posible que la acción ABRE no vaya acompañada de un objeto asociado.¿Que objeto acompañará a la acción?: ABRE` y se queda esperando hasta que reciba una respuesta concreta del usuario. Inmediatamente después de leer este mensaje, el usuario tecleará el objeto que ha olvidado introducir y el sistema completa la frase con dicho objeto. Finalmente se reconoce la frase `ABRE PUERTA` que es lo que el usuario quería decir.

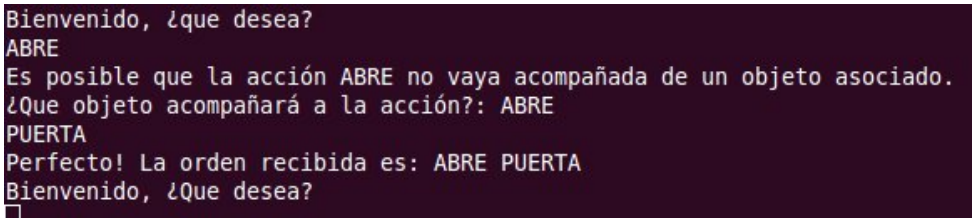


Figura 4.5: Ejemplo de omision de objeto.

Como hemos visto, el sistema es capaz de detectar omisiones de objetos en sentencias simples. A con-

tinuación se presenta el caso de un usuario que emite una sentencia simple sin la acción correspondiente, es decir, un único objeto.

4.4.1.2 Omisión del acción

En el ejemplo de la figura 4.6, el usuario a emitido la sentencia `PUERTA`, sentencia a la que le falta la acción que llevará asociado este objeto. El sistema detecta esta situación y pregunta al sistema por dicha acción por medio de la cuestión: `Es posible que el objeto PUERTA no vaya acompañado de una acción asociada. Por favor, exprese la acción que se ejecutará sobre el objeto: PUERTA`. El usuario ante la información proporcionada por el sistema responde con la acción que deseaba aplicar sobre la `PUERTA` que es `CIERRA`, resolviendo la ambigüedad.

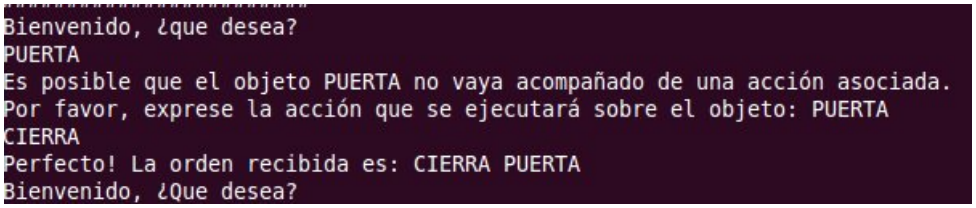


Figura 4.6: Ejemplo de omisión de acción.

4.4.2 Errores en sentencias complejas

Antes de empezar a mostrar el comportamiento del sistema de diálogo debemos recordar que una sentencia compleja o compuesta es aquella que está formada por dos o más sentencias simples, es decir, una sentencia en donde existan al menos dos acción y dos objetos asociados a cada una de ellas. En este tipo de sentencias se pueden producir varios tipos de errores. Veremos como el sistema es capaz de detectar cada uno de ellos y resolverlos.

4.4.2.1 Acción y objeto no relacionados

Este tipo de sentencias erróneas se producen cuando el usuario introduce un acción y el objeto que la sigue no está relacionado con ella en ningún sentido. Este tipo de error se suele cometer debido a la rapidez en el habla. Normalmente se origina cuando el usuario quiere emitir una sentencia compleja con dos acciones y dos objetos y junto la acción de la primera con el objeto de la segunda. En la figura 4.7 se muestra un error de este tipo. El usuario introduce la sentencia `CLOSE LAMP` frase sin sentido alguno. El sistema es capaz de detectar este error suponiendo que el usuario a querido decir una sentencia compleja formada por dos sentencias simples. A la primera le faltaría el objeto y a la segunda la acción.

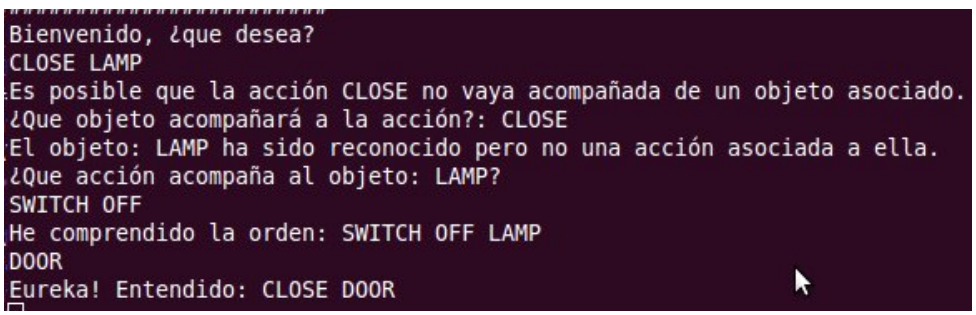


Figura 4.7: Ejemplo de acción y objeto no relacionados.

Como vemos en la figura 4.7 el sistema detecta la incoherencia y pregunta por el objeto asociado a la acción `CLOSE` y por la acción del objeto `LAMP`.

4.4.2.2 Doble acción sin objetos relacionados

Este caso se da cuando el usuario emite dos o mas acciones sin sus correspondientes objetos asociados. En la imagen de la figura 4.8 se observa como el usuario a emitido la sentencia `CLOSE AND TURN` con dos acciones sin sus correspondientes objetos asociados. El sistema responde preguntando por dichos objetos al usuario, suponiendo que la sentencia compleja está formada por dos sentencias simples.

```
Bienvenido, ¿que desea?
CLOSE TURN
Es posible que la acción CLOSE no vaya acompañada de un objeto asociado.
¿Que objeto acompañará a la acción?: CLOSE
La acción: TURN ha sido reconocida pero no un objeto asociado a ella.
¿Que objeto acompaña a la acción: TURN?
DOOR ROBOT
He comprendido la orden: CLOSE DOOR
Eureka! Entendido: TURN ROBOT
```

Figura 4.8: Ejemplo de doble acción sin objetos relacionados.

En este caso, el sistema de diálogo pregunta al usuario por los objetos asociados a las acciones `CLOSE` y `TURN` y el usuario responde con los correspondientes objetos que se indican en la figura 4.8.

Como vemos, hasta ahora el sistema de diálogo cumple con la detección de errores y con la corrección de los mismos. Vamos a ver otro ejemplo de sentencia compleja errónea.

4.4.2.3 Doble objeto sin acciones asociadas

Se trata del caso inverso al anterior. Aquí, el usuario a emitido una sentencia con dos objetos sin sus correspondientes acciones que aplicar sobre ellos. Al igual que el error anterior, estos casos se produce normalmente debido a la rapidez del habla la cual provoca la mezcla de dos sentencia simples. En la figura 4.9 se muestra una sentencia emitida por el usuario de este tipo. El sistema de diálogo recibe la sentencia `ROBOT AND DOOR`, dos objetos sin acción asociada.

```
Bienvenido, ¿que desea?
ROBOT DOOR
Es posible que el objeto DOOR no vaya acompañado de una acción asociada.
Por favor, exprese la acción que se ejecutará sobre el objeto: DOOR
El objeto: ROBOT ha sido reconocido pero no una acción asociada a ella.
¿Que acción acompaña al objeto: ROBOT?
CLOSE
He comprendido la orden: CLOSE DOOR
TURN
Eureka! Entendido: TURN ROBOT
```

Figura 4.9: Ejemplo de doble objeto sin acciones relacionadas.

Como vemos, la situación es detectada y el usuario introduce las correspondientes acciones esta vez una a una porque así lo ha querido. El sistema de diálogo puede detectar también esta situación errónea y corregir solicitando información al usuario como venía haciendo en los casos anteriores.

4.5 El algoritmo *blackboard*

Supongamos que introducimos a nuestro sistema una sentencia simple con el objeto de la sentencia omitido, como por ejemplo `CLOSE`. El algoritmo del sistema de diálogo, como vimos en la sección 3.6 del capítulo 3, pone la acción en la *blackboard* de tal modo que pueda ser resuelta en sucesivas iteraciones. En el mismo momento en el que se introduce en la *blackboard* un elemento, se ele asigna una edad. Cada vez que se ejecute una iteración del algoritmo y el elemento no se haya extraído de la misma, se incrementará su edad en una unidad. Si se rebasa la edad límite para permanecer en la *blackboard*, el elemento será descartado. Este es el funcionamiento principal del algoritmo *blackboard*, técnica que utiliza el algoritmo del sistema de diálogo para resolver la mayor parte de los errores explicados con anterioridad.

En las figuras 4.10, 4.11, 4.12 y 4.13, se muestra una serie de iteraciones sucesivas del sistema de diálogo que van introduciendo elementos en la *blackboard* con el objetivo de ser resueltos.

La figura 4.10 muestra la primera iteración del algoritmo en el que el usuario emite la sentencia `TURN` y se almacena en `blackboard`. Como se puede observar, la *blackboard* contiene el elemento `TURN` y la asigna una edad 0.

Figura 4.10: Iteración 1.

La figura 4.11 muestra la segunda iteración. El usuario emite la sentencia `CLOSE` que también se añade a la `blackboard`. Además se incrementa la edad de los elementos en la `blackboard`.

La figura 4.12 muestra la tercera iteración. El usuario emite la sentencia `LAMP` que también se añade a la `blackboard`. Además se incrementa la edad de los elementos en la `blackboard`. Como se observa en la imagen, la acción `TURN` tiene edad 2, la acción `CLOSE` tiene edad 1 y el objeto `LAMP` tiene edad 0. Vemos como se van incrementan las edades de los elementos en la *blackboard*.

La figura 4.13 muestra la cuarta iteración. El usuario emite la sentencia `DOOR` para enlazarla con la acción `CLOSE` almacenada en la `blackboard`. En este caso lo que ocurre es que se extrae esta acción de la `blackboard` y se incrementa la edad del resto de los elementos. La sentencia `CLOSE DOOR` es comprendida con éxito-.

Como vemos, el algoritmo *blackboard* almacena las sentencias incorrectas parcialmente comprendidas para que puedan ser resueltas en futuras iteraciones como es el anterior caso.

A continuación, vamos a mostrar como el algoritmo del sistema de diálogo elimina los componentes

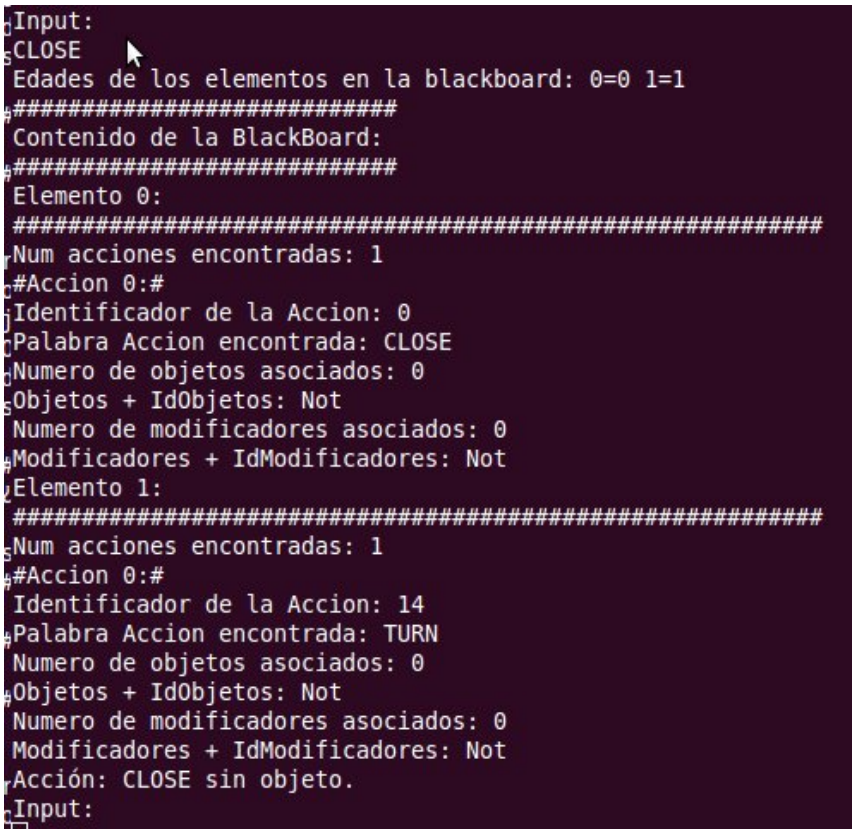


Figura 4.11: Iteración 2.

de la *blackboard* con edad superior a 5. Para ello, vamos a insertar un elemento en la *blackboard* y vamos a introducir sentencias correctas hasta que sea eliminado de la misma.

En la figura 4.14 se muestra el resultado de la primera iteración.

En la figura 4.15 se muestra el resultado de la segunda iteración.

En la figura 4.16 se muestra el resultado de la tercera iteración.

En la figura 4.17 se muestra el resultado de la primera iteración.

En la figura 4.18 se muestra el resultado de la quinta iteración.

En la figura 4.19 se muestra el resultado de la sexta iteración. Es en esta iteración donde finalmente se elimina el elemento de la *blackboard* al alcanzar la edad límite 5.

4.6 La técnica de cancelación

En esta sección vamos a mostrar de qué manera funciona la técnica de cancelación que vimos en el capítulo 3 en la sección 3.7. La técnica de cancelación se puede utilizar en varios contextos diferentes. Uno de ellos puede ser uno en el que usuario haya emitido correctamente una sentencia y se arrepienta inmediatamente después de haberla emitido. Otro caso puede ser uno en el que el usuario haya emitido una sentencia incorrecta por equivocación y quiera rectificar independientemente de que el sistema trate de corregir el error. Aquí debemos recordar que muchos de los errores con los que el sistema de diálogo deberá dialogar son heredados de los sistemas de reconocimiento y que por lo tanto, debido a diversas causas pueden estar transcribiendo las sentencias incorrectas.

```
Input:
LAMP
Edades de los elementos en la blackboard: 0=0 1=1 2=2
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: -1
Numero de objetos asociados: 1
Objetos + IdObjetos: LAMP(0) ->
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Elemento 1:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 0
Palabra Accion encontrada: CLOSE
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Elemento 2:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Objeto LAMP sin acción.
Input:
```

Figura 4.12: Iteración 3.

```
Input:
DOOR
Edades de los elementos en la blackboard: 0=0 1=1 2=2 3=3
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 0
Elemento 1:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 0
Palabra Accion encontrada: CLOSE
Numero de objetos asociados: 1
Objetos + IdObjetos: DOOR(0) ->
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Elemento 2:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Eureka! Entendido: CLOSE DOOR
```

Figura 4.13: Iteración 4.

```
Bienvenido, ¿que desea?
TURN
Edades de los elementos en la blackboard: 0=0
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Acción: TURN sin objeto.
Input:
```

Figura 4.14: Iteración 1.

```
TURN ROBOT
Edades de los elementos en la blackboard: 0=1
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Eureka! Entendido: TURN ROBOT
```

Figura 4.15: Iteración 2.

```
CLOSE THE DOOR
Edades de los elementos en la blackboard: 0=2
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Perfecto! La orden recibida es: CLOSE DOOR
```

Figura 4.16: Iteración 3.

```
TURN ROBOT
Edades de los elementos en la blackboard: 0=3
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
He comprendido la orden: TURN ROBOT
```

Figura 4.17: Iteración 4.

```
CLOSE DOOR
Edades de los elementos en la blackboard: 0=4
############################
Contenido de la BlackBoard:
############################
Elemento 0:
############################################################
Num acciones encontradas: 1
#Accion 0:#
Identificador de la Accion: 14
Palabra Accion encontrada: TURN
Numero de objetos asociados: 0
Objetos + IdObjetos: Not
Numero de modificadores asociados: 0
Modificadores + IdModificadores: Not
Eureka! Entendido: CLOSE DOOR
```

Figura 4.18: Iteración 5.

```
OPEN DOOR
Edades de los elementos en la blackboard:
############################
Contenido de la BlackBoard:
############################
He comprendido la orden: OPEN DOOR
```

Figura 4.19: Iteración 6.

En el ejemplo de la figura 4.20 puede observarse como el usuario emitió una sentencia perfectamente correcta (`TURN ROBOT`) y como el usuario rectifico después emitiendo la orden `CANCEL SENTENCE`. El sistema de diálogo comprende la sentencia recibida y la ejecuta. Posteriormente, al recibir la orden de cancelación, el sistema de diálogo busca la última sentencia correcta comprendida y elimina su efecto.

```
Bienvenido, ¿que desea?
TURN ROBOT
Eureka! Entendido: TURN ROBOT
CANCEL SENTENCE
Perfecto! La orden recibida es: CANCEL SENTENCE
Cancelada el comando anterior: TURN ROBOT
```

Figura 4.20: Ejemplo de sentencia correcta cancelada.

A continuación vamos a ver otra situación en donde el uso de la técnica de cancelación puede ser muy útil. El usuario emite una sentencia errónea equivocada como por ejemplo la sentencia `ABRE` cuando lo que quería decir era `CIERRA LA PUERTA`. La figura 4.21 muestra este ejemplo. El usuario se da cuenta del error que ha cometido al introducir la sentencia `ABRE` y decide emitir la sentencia de cancelacion `CANCELAR SENTENCIA` para borrar la acción `ABRE` de la *blackboard*.

Al cancelar la sentencia errónea estamos evitando que se almacene información en la *blackboard* que después, en sucesivas iteraciones puedan entrar en conflicto con otras futuras correcciones, incluso habiendo introducido el mecanismo de envejecimiento.

```
Bienvenido, ¿que desea?
ABRE
Es posible que la acción ABRE no vaya acompañada de un objeto asociado.
¿Que objeto acompañará a la acción?: ABRE
CANCELAR SENTENCIA
Perfecto! La orden recibida es: CANCELAR SENTENCIA
Borrada la BlackBoard
CIERRA LA PUERTA
He comprendido la orden: CIERRA PUERTA
```

Figura 4.21: Ejemplo de sentencia incorrecta cancelada.

4.7 El generador de respuestas

Vimos en la sección 3.8 del capítulo 3 que el generador de respuestas construía las sentencias que debía emitir al usuario a partir de dos parámetros fundamentales: el modo de funcionamiento y el aleatorizador de respuestas. A continuación se van a poner varios ejemplos del funcionamiento de los dos elementos y de su efecto:

4.7.1 El modo de uso

A continuación vamos a ver como responde el sistema ante una sentencia simple a la que le falta el objeto sobre el que aplicar la acción. En la figura 4.22 se muestra un ejemplo de como el sistema genera sentencias al usuario para preguntarle por el objeto asociado a la acción `TURN` en el modo cadete. En este modo se da mucha información para que el usuario sepa de forma exacta la razón por la que se ha cometido error. El modo cadete es el modo que debe utilizar un usuario inexperto.

```
Bienvenido, ¿que desea?
TURN
Es posible que la acción TURN no vaya acompañada de un objeto asociado.
¿Que objeto acompañará a la acción?: TURN
```

Figura 4.22: Ejemplo del modo cadete.

En la figura 4.23 se muestra la misma situación que la de la figura 4.22 pero el generador de respuestas está configurado en modo veterano. En este modo se da la información necesaria de los errores que se están cometiendo, sin entrar en mucho detalle. Está diseñado para usuarios expertos del sistema.

```
Bienvenido, ¿que desea?
TURN
Acción: TURN sin objeto.
```

Figura 4.23: Ejemplo del modo veterano.

4.7.2 El aleatorizador de respuestas

Vimos en la sección 3.8.2 del capítulo 3 que el aleatorizador de respuestas escogía aleatoriamente una respuesta de entre un conjunto de respuestas equivalentes. De esta manera dábamos una aspecto más humano a la conversación mantenida con el usuario.

En la figura 4.24 se muestran varias de estas sentencias respuestas en diferentes iteraciones para que se pueda apreciar la labor del aleatorizador. Observe como las respuestas que proporciona el sistema ante la misma sentencia no son idénticas pero sí equivalentes. Algunas se repiten puesto que aleatoriamente se selecciona una del conjunto.

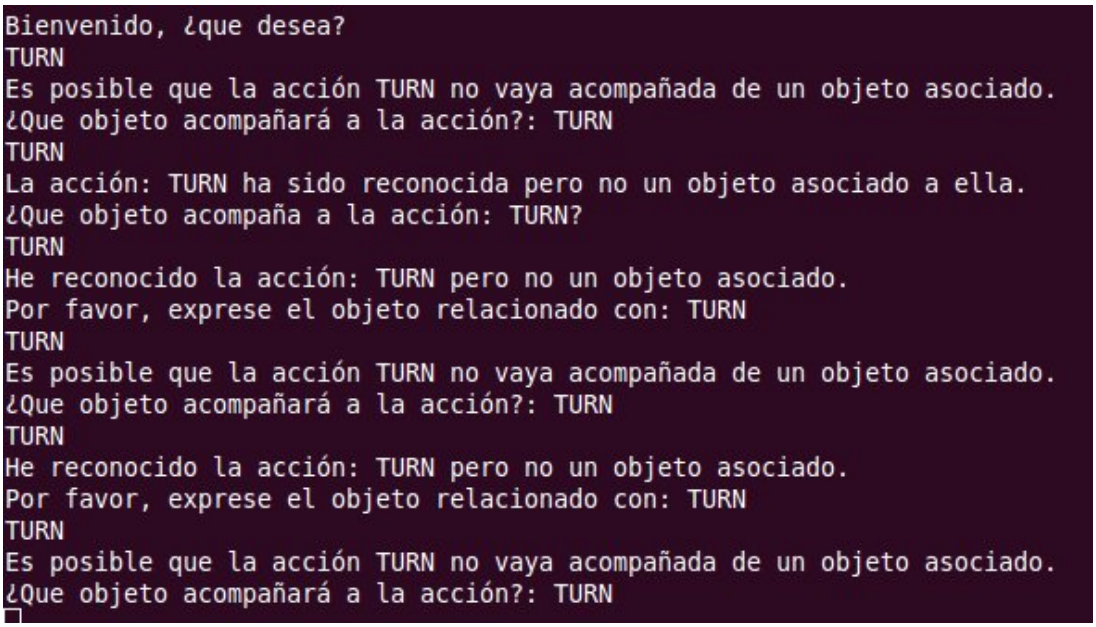


Figura 4.24: Ejemplo del aleatorizador de respuestas. Modo cadete.

Para el caso de un usuario veterano, la sentencia que se emite es tan escueta que no se ha implementado esta opción. Véase la figura 4.25.

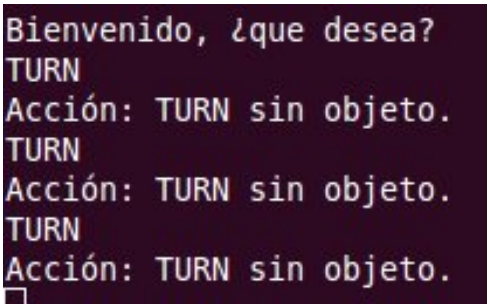


Figura 4.25: Ejemplo del aleatorizador de respuestas. Modo veterano.

4.8 Conclusiones

De todo lo expuesto en las secciones anteriores, podemos concluir que el sistema de diálogo diseñado es capaz de comprender aquellas sentencias para las cuales el sistema fue diseñado y detectar los principales errores en sentencias simples y compuestas. El sistema no solamente es capaz de detectar errores en las sentencias recibidas sino que también está provisto de mecanismos de corrección que por medio de la interrogación del usuario son capaces de comprender las sentencias erróneas. También hemos visto en este capítulo el funcionamiento del algoritmo *blackboard*, y las técnicas de cancelación y de generación de respuestas.

Capítulo 5

Conclusiones y líneas futuras

La conversación es la expresión de nuestro modo de pensar.

Séneca

5.1 Introducción

En este apartado se resumen las conclusiones obtenidas a lo largo de este proyecto final de máster y se proponen futuras líneas de investigación que se deriven del trabajo y alguna que otra mejora.

El capítulo se estructura en dos secciones principales:

- La primera sección dará cuenta de las principales conclusiones obtenidas del proyecto realizado. En esta sección, haremos una valoración de si realmente se han logrado alcanzar los objetivos propuestos.
- La segunda sección mostrará las líneas futuras de este proyecto. En este apartado se expondrán algunas ideas que podrían mejorar aun más el comportamiento del sistema de diálogo de cara al usuario.

5.2 Conclusiones

Haciendo revisión de cada uno de los objetivos que describimos en el apartado 1.4, podemos concluir que hemos logrado alcanzar la meta que nos impusimos con este proyecto: diseñar, implementar y evaluar un sistema de diálogo completo, con un grado elevado de naturalidad. Como nos hemos ido dando cuenta a lo largo de la exposición de este trabajo, un sistema de diálogo es un sistema muy complejo compuesto de varios puntos cada uno de ellos con su problemática asociada. Haber alcanzado un sistema de diálogo funcional y operativo diseñado para el espacio inteligente que describimos en la introducción ha supuesto todo un reto tanto personal como profesional. El sistema de diálogo construido es capaz de interpretar, comprender las sentencias recibidas, bien por parte de los reconocedores acoplados a la entrada, bien por parte del propio usuario que introduce la sentencias ya transcritas, y hacer frente con éxito a un buen número de fenómenos propios de la interacción humana.

A continuación se va a exponer una serie de puntos que constituyen los logros que hemos obtenido clasificándolos en función del módulo funcional que seguimos para la construcción del sistema de diálogo completo:

Vocabulario y gramática: Hemos sido capaces de diseñar un vocabulario y una gramática relacionada con el contexto del interfaz de control multimodal del espacio inteligente que describimos en la introducción de este proyecto. La forma en que hemos clasificado el vocabulario en acciones, modificadores, y objetos permite una fácil extrapolación del sistema a cualquier otro espacio inteligente que nosotros diseñemos. Además, la manera en la que el sistema de diálogo asocia un número identificador y un nombre a todas aquellas palabras que representen el mismo símbolo tiene una ventaja añadida: Podemos relacionar conjuntos de palabras que procedan de distintos reconocedores o conjuntos de palabras pertenecientes a distintos idiomas que posean el mismo significado. Esta propiedad a hecho posible que el sistema de diálogo diseñado pueda recibir sentencias tanto del reconocedor de gesto como del reconocedor de voz, este ultimo con salida en dos idiomas(inglés y español). A modo de ejemplo las palabras `CLOSE`, `ABRE` y `ABREg` procedentes del reconocedor de voz en ingles y en español y del reconocedor de gestos, poseen el mismo significado semántico y, por lo tanto, el sistema de diálogo les asocia un único identificador y una palabra representante del grupo. Esta es la manera que tiene el sistema de diálogo de abstraerse de los distintos reconocedores e idiomas y así extraer el significado semántico de las sentencias.

Errores: Se ha logrado desarrollar una lista de los errores más comunes que se pueden dar en un espacio de este tipo. Estos errores son comunes sobre todo cuando se trabaja con el lenguaje oral. Nuestro sistema de diálogo es capaz de resolver los errores más comunes que en una conversación se pueden dar. En el capítulo 4 (resultados) se muestra como se han conseguido resolver todos los posibles errores que un usuario común puede cometer a la hora de dirigirse al sistema, incluso algunos no tan comunes.

Estructuración de los datos: El sistema de almacenaje de la información es lo suficientemente abstracto como para que pueda ser utilizado en cualquier otro espacio inteligente. Las estructura de datos de almacenaje permite guardar distintos tipos de vocabulario siempre que se pueda clasificar en acciones, objetos y modificadores. Esta forma de estructurar el sistema permite a futuros programadores utilizar el programa donde se almacena el sistema de diálogo para adaptarlo a otros tipos de contexto, en otros espacios inteligentes con otro vocabulario diferente. Este hecho es uno de los grandes hitos del proyecto. Haber sabido generar un sistema de diálogo general y fácilmente adaptable a otros tipos de escenarios es un gran logro a reseñar.

Algoritmo: Uno de las labores fundamentales que hemos logrado con éxito ha sido la confección de un algoritmo que permita comprender las sentencias correctas que recibe el sistema de diálogo a su entrada, detectar posibles errores en las mismas y corregir aquellas en las que se hayan detectado dichos errores. El uso de una técnica basada en memoria como lo es el algoritmo *blackboard* ha conferido al sistema establecer diálogos más humanos. Como vimos en la capítulo 3 (desarrollo), el algoritmo es lo suficientemente complicado como para resolver toda la casuística de errores que se pueden dar. Desarrollar un algoritmo de estas dimensiones es un trabajo que ha requerido varios meses. Son muchas las condiciones que pueden o no cumplirse y se deben garantizar el funcionamiento para todos los casos. Este algoritmo cumple tal propósito y como tal debe destacarse su buen funcionamiento.

Técnica de cancelación La técnica de cancelación ha proferido un singular mejora al permitir al usuario cancelar cualquier acción previamente introducida. Este hecho proporciona gran versatilidad al usuario al poder rectificar incluso habiendo introducido sentencias correctas al sistema.

Generador de respuestas: En este punto nos hemos sorprendido de los grandes avances que a nuestro sistema de diálogo proporciona la introducción de los modos de funcionamiento, y del aleatorizador

de respuestas, sobre todo en lo relativo a la consecución de unos diálogos menos robotizados, más humanos.

Has ahora hemos abordado las conclusiones a las que hemos llegado una vez completado nuestro proyecto. A continuación se expondrán algunas mejoras y lineas futuras que podemos implementar sobre el proyecto que hoy está construido.

5.3 Líneas futuras

A lo largo del proyecto muchas son las ideas que se nos han ido ocurriendo para mejorar el sistema de diálogo. La mayoría de ellas no se han podido realizar, bien por falta de tiempo, bien porque su desarrollo implica un trabajo de investigación que se sale de los objetivos de este proyecto fin de máster. Las principales lineas futuras que se podrían.

Modelo estadístico del algoritmo: El planteamiento del sistema de diálogo que hemos diseñado es determinista, es decir, nosotros algoritmo comprueba que la sentencia recibida se ajuste a un modelo que almacenamos en una base de datos y que básicamente debe cumplir unas reglas perfectamente definidas. Una perspectiva basada en probabilidades definiría un modelo estadístico como redes bayesianas o redes markovianas y trataría de buscar la sentencia más probable de entre todas las posibles dentro de dicho modelo. Se demuestra empíricamente que estos modelos poseen un mejor resultado que los deterministas cuando se incrementa el tamaño del vocabulario y se complican las reglas de la gramática. En un futuro, se podría sustituir el algoritmo determinista diseñado por uno basado en probabilidades.

Aumento del vocabulario y de las reglas de la gramáticas: Una forma de mejorar el sistema y hacerlo más cómodo al usuario y versátil sería aumentar tanto el tamaño del vocabulario que maneja como el número de reglas que conforma la gramática. Al igual que una persona más culta posee mucha riqueza de vocabulario y de gramática, los sistemas de diálogo mejoran al incrementar estos parámetros.

Aumento de los modos o roles El aumento de los modos o roles que el generados de funciones puede manejar incrementaría la funcionalidad del sistema de diálogo. Actualmente tenemos implementado el modo cadete y el modo veterano. Quizá un modo intermedio mejoraría las prestaciones del sistema.

Portabilidad total a $C++$: El sistema de diálogo ha sido diseñado en dos lenguajes de programación diferentes C y $C++$ por retrocompatibilidad. Exportar todo el sistema a $C++$ sería algo deseable aun cuando los dos lenguajes son compatibles entre ellos.

Bibliografía

[1] M. Villaverde Díez, "Diseño, implementación y evaluación de una interfaz de control multimodal en un espacio inteligente: control vocal," Master's thesis, Escuela Politécnica Superior. Universidad de Alcalá. Spain, 2013.

[2] D. Casillas Pérez, "Diseño, implementación y evaluación de una interfaz de control multimodal en un espacio inteligente: control gestual," Master's thesis, Escuela Politécnica Superior. Universidad de Alcalá. Spain, 2013.

[3] "Información acerca de las interfaces hombre-máquina," http://es.wikipedia.org/wiki/Interacción_persona-computador [último acceso noviembre 2013].

[4] "Imagen de un ratón," http://www.mundopc.es/udecontrol_datos/objetos/86.jpg [último acceso noviembre 2013].

[5] "Imagen de un teclado," http://www.curiosidadesenlared.com/wp-content/uploads/2013/05/teclado.jpg [último acceso noviembre 2013].

[6] "Imagen de un interfaz inteligente, extraida de la pelicula *Minority Report*," http://www.dailygalaxy.com/photos/uncategorized/2008/02/27/minorityreport.jpg [último acceso noviembre 2013].

[7] "Imagen del sensor Kinect (Xbox LIVE)," http://desarrolladoresdevideojuegos.es/2011/08/08/\T1\textquestiondowncomo-empezar-a-desarrollar-con-kinect [último acceso noviembre 2013].

[8] "Imagen del microfono *close-talk Logitech H760*," http://hardzone.es/2010/08/27/logitech-wireless-headset-h760-nuevos-auriculares-inalambricos/ [último acceso nombiembre 2013].

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[10] K.-F. Lee, "Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 4, pp. 599–609, 1990.

[11] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[12] F. de Saussure, "Curso de lingüística general." 1913.

[13] "Página de información sobre sistemas de diálogo y su tipología," http://liceu.uab.es/~joaquim/speech_technology/tecnol_parla/dialogue/dialogo_general/sistemas_dialogo.html#tipologia_sistemas_dialogo [último acceso septiembre 2014].

[14] F. Fernández-MartíNez, J. Ferreiros, J. M. Lucas-Cuesta, J. M. Montero-MartíNez, R. San-Segundo, and R. Córdoba, "Towards building intelligent speech interfaces through the use of more flexible, robust and natural dialogue management solutions," *Interact. Comput.*, vol. 24, no. 6, pp. 482–498, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.intcom.2012.09.003

[15] F. Fernández-Martínez, J. Ferreiros, J. M. Lucas-Cuesta, J. D. Echeverry, R. San-Segundo, and R. Córdoba, "Flexible, robust and dynamic dialogue modeling with a speech dialogue interface for controlling a hi-fi audio system," in *Proceedings of the IEEE Workshop on Database and Expert Systems Applications (DEXA 2010)*. Bilbao, Spain: Springer, September 2010.

[16] F. F. Martinez, J. Ferreiros, R. Cordoba, J. M. Montero, R. San-Segundo, and J. M. Pardo, "A bayesian networks approach for dialog modeling: The fusion bn," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ser. ICASSP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 4789–4792. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2009.4960702

[17] "Información sobre GNU/Linux en wikipedia," http://es.wikipedia.org/wiki/GNU/Linux [último acceso noviembre 2013].

[18] "Página de la aplicación Emacs," http://savannah.gnu.org/projects/emacs/ [último acceso noviembre 2013].

[19] "Página de la aplicación KDevelop," http://www.kdevelop.org [último acceso noviembre 2013].

[20] L. Lamport, *LaTeX: A Document Preparation System, 2nd edition.* Addison Wesley Professional, 1994.

[21] "Página de la aplicación Octave," http://www.octave.org [último acceso noviembre 2013].

[22] "Página de la aplicación CVS," http://savannah.nongnu.org/projects/cvs/ [último acceso noviembre 2013].

[23] "Página de la aplicación GCC," http://savannah.gnu.org/projects/gcc/ [último acceso noviembre 2013].

[24] "Página de la aplicación make," http://savannah.gnu.org/projects/make/ [último acceso noviembre 2013].

Apéndice A

Herramientas y recursos

Las herramientas necesarias para la elaboración del proyecto han sido:

- PC compatible
- Sistema operativo GNU/Linux [17]
- Entorno de desarrollo Emacs [18]
- Entorno de desarrollo KDevelop [19]
- Procesador de textos LaTeX[20]
- Lenguaje de procesamiento matemático Octave [21]
- Control de versiones CVS [22]
- Compilador C/C++ gcc [23]
- Gestor de compilaciones make [24]

Apéndice B

Vocabulario y Gramática del sistema de diálogo

B.1 Introducción

El objetivo de este apéndice es proporcionar al usuario toda la información relativa al vocabulario que el sistema de diálogo maneja así como de las reglas que conforman la gramática. Este apéndice es un glosario de todas las palabras y sentencias simples que un usuario ideal puede conformar. Por supuesto, las sentencias simples pueden concatenarse, así que las posibles sentencias que puede emitir un usuario son infinitas. Aquí solamente se muestran las posibles sentencia simples que se pueden formar.

B.2 Descripción del vocabulario y de la gramática

A continuación se presenta organizado por acciones tanto la gramática como el vocabulario de todo el sistema de diálogo. Faltaría por añadir la palabra clave `CANCELAR` como señal de cancelación de las operaciones anteriores. Además se añade el efecto que se produciría en el espacio inteligente virtual sobre el que se ejecutan las ordenes comandadas por el usuario.

B.2.1 Call

- Acciones:
  - CALL
  - LLAMA
  - LLAMAR
  - PHONE
- Objetos:
  - SARA ALARCON
  - ANTHONY WATT
  - BENJAMIN FRANKLIN
  - IGNACIO BOLONIO

  - JAMES BROWN
  - CARL JOHNSON
  - DAVID CASILLAS
  - MORGAN FREEMAN
  - PETER GRIFFIN
  - GUILLE VIDALES
  - GUILLERMO VIDALES
  - LIDIA LOPEZ
  - MANU VILLAVERDE
  - MANUEL VILLAVERDE
  - MARTA MARRON
  - PAUL NEWMAN
  - PASCU
  - DAVID PASCUAL
  - ROBERT REDFORD
  - ROBIN WILLIAMS
  - ALARCON
  - WATT
  - FRANKLIN
  - BOLONIO
  - BROWN
  - GRIFFIN
  - VIDALES
  - MACIAS
  - LOPEZ
  - VILLAVERDE
  - MARRON
  - NEWMAN
  - MARTA MARRON
  - PASCUAL
  - REDFORD
  - WILLIAMS

- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - No produce cambios en el espacio virtual.
- Mensajes:
  - Objetos encontrados: Mensaje de información avisando de que se llamará a los usuarios seleccionados.
  - Objetos no encontrados: Mensaje de error avisando de la falta de objetos.

B.2.2 Close

- Acciones:
  - CIERRA
  - CLOSE
- Objetos:
  - PUERTA | DOOR
  - VENTANA | WINDOWS
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si se encuentra un objeto adecuado cuyo estado sea abierto, se cerrará.
- Mensajes:
  - Objeto encontrado / Estado abierto: Mensaje de información avisando de que el objeto se ha cerrado.
  - Objeto encontrado / Estado cerrado: Mensaje de error, no se puede cerrar el objeto.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.3 Come

- Acciones:
  - COME
  - VEN
- Objeto:
  - ROBOT
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si se encuentra el objeto robot, este se acudirá a la posición del avatar del usuario.
- Mensajes:
  - Objeto encontrado: Mensaje de información avisando de que el robot se está acercando.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.4 Dial

- Acciones:
  - DIAL
  - MARCA
- Objetos:
  - CERO
  - UNO
  - DOS
  - TRES
  - CUATRO
  - CINCO
  - SEIS
  - SIETE
  - OCHO
  - NUEVE
  - ZERO
  - OH
  - ONE
  - TWO
  - THREE
  - FOUR
  - FIVE
  - SIX
  - SEVEN
  - EIGHT
  - NINE
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - No produce cambios en el espacio virtual.
- Mensajes:
  - Objetos encontrados: Mensaje de información avisando de que se llamará al número seleccionado.
  - Objetos no encontrados: Mensaje de error avisando de la falta de objetos válidos.

B.2.5 Down

- Acciones:
  - BAJA
  - DOWN
  - LOWER
- Objetos:
  - BLINDS
  - PERSIANAS
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si encuentra el objeto persiana y este se encuentra subido, lo bajará.
- Mensajes:
  - Objeto encontrado / Estado subido: Mensaje de información avisando de que se bajarán las persianas.
  - Objeto encontrado / Estado bajado: Mensaje de error, no se puede bajar las persianas.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.6 Go

- Acciones:
  - GO
  - VE
- Objeto:
  - ROBOT
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si encuentra el objeto robot lo desplazará a una posición predeterminada alejada del avatar del usuario.
- Mensajes:
  - Objeto encontrado: Mensaje de información avisando de que el robot procederá a desplazarse al punto predeterminado.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.7 Go Back

- Acciones:
  - GO BACK
  - RETROCEDE
- Objeto:
  - ROBOT
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si encuentra el objeto robot, este procederá a desplazarse 10 cm en sentido contrario al que indique su dirección actual.
- Mensajes:
  - Objeto encontrado: Mensaje de información avisando de que el robot procederá a retroceder.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.8 Go On

- Acciones:
  - AVANZA
  - GO ON
- Objeto:
  - ROBOT
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si encuentra el objeto robot, este procederá a desplazarse 10 cm en el mismo sentido al que indique su dirección actual.
- Mensajes:
  - Objeto encontrado: Mensaje de información avisando de que el robot procederá a avanzar.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.9 Less

- Acciones:
  - LESS
  - MENOS
- Objeto:
  - LUZ | LIGHT
  - SONIDO | SOUND
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Objeto LUZ | LIGHT:
    - Si el objeto cumple las condiciones necesarias, disminuimos la transparencia de la luz del techo un 10 %.
  - Objeto SONIDO | SOUND:
    - No produce cambios en el espacio visual.
- Mensajes:
  - Objeto LUZ | LIGHT:
    - Luz del techo encendida / Luz mayor del 0 %: Mensaje informando de que la luz ha disminuido un 10 %.
    - Luz del techo encendida / Luz al 0 %: Mensaje de error indicando que no se puede bajar más la luz.
    - Luz del techo apagada: Mensaje de error indicando que debe encender primero la lampara principal.
  - Objeto SONIDO | SOUND:
    - Radio encendida / Sonido mayor del 0 %: Mensaje informando de que el volumen ha disminuido un 10 %.
    - Radio encendida / Sonido al 0 %: Mensaje de error indicando que no se puede bajar más la el sonido.
    - Radio apagada: Mensaje de error indicando que debe encender primero la radio.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.10 More

- Acciones:
  - MAS
  - MORE
- Objeto:

  - LUZ | LIGHT
  - SONIDO | SOUND

- Modificadores:
  - No tiene.

- Efectos en el espacio virtual:
  - Objeto LUZ | LIGHT:
    * Si el objeto cumple las condiciones necesarias, aumentamos la transparencia de la luz del techo un 10 %.
  - Objeto SONIDO | SOUND:
    * No produce cambios en el espacio visual.

- Mensajes:
  - Objeto LUZ | LIGHT:
    * Luz del techo encendida / Luz menor del 100 %: Mensaje informando de que la luz ha aumentado un 10 %.
    * Luz del techo encendida / Luz al 100 %: Mensaje de error indicando que no se puede aumentar más la luz.
    * Luz del techo apagada: Mensaje de error indicando que debe encender primero la lampara principal.
  - Objeto SONIDO | SOUND:
    * Radio encendida / Sonido menor del 100 %: Mensaje informando de que el volumen ha aumentado un 10 %.
    * Radio encendida / Sonido al 100 %: Mensaje de error indicando que no se puede aumentar más la el sonido.
    * Radio apagada: Mensaje de error indicando que debe encender primero la radio.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.11 Open

- Acciones:
  - ABRE
  - OPEN

- Objetos:
  - PUERTA | DOOR
  - VENTANA | WINDOWS

- Modificadores:
  - No tiene.

- Efectos en el espacio virtual:

  - Si se encuentra un objeto adecuado cuyo estado sea cerrado, se abrirá.

- Mensajes:
  - Objeto encontrado / Estado cerrado: Mensaje de información avisando de que el objeto se ha abierto.
  - Objeto encontrado / Estado abierto: Mensaje de error, no se puede abrir el objeto.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.12 Raise

- Acciones:
  - PULL UP
  - RAISE
  - SUBE
- Objetos:
  - BLINDS
  - PERSIANAS
- Modificadores:
  - No tiene.
- Efectos en el espacio virtual:
  - Si encuentra el objeto persiana y este se encuentra bajado, lo subirá.
- Mensajes:
  - Objeto encontrado / Estado bajado: Mensaje de información avisando de que se subirán las persianas.
  - Objeto encontrado / Estado subido: Mensaje de error, no se puede subir las persianas.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.13 Turn

- Acciones:
  - GIRA
  - GIRA_DERECHA[1].
  - GIRA_IZQUIERDA
  - TURN
- Objeto:
  - ROBOT

[1]Tanto GIRA_DERECHA como GIRA_IZQUIERDA se consideran verbos y modificadores al mismo tiempo y no es necesario incluirlos dos veces

- Modificadores:
  - DERECHA | RIGHT | GIRA_DERECHA
  - IZQUIERDA | LEFT | GIRA_IZQUIERDA
- Efectos en el espacio virtual:
  - DERECHA | RIGHT | GIRA_DERECHA: Si encuentra el objeto robot, este procederá a girar la dirección a la que apunta 90º en sentido horario.
  - IZQUIERDA | LEFT | GIRA_IZQUIERDA: Si encuentra el objeto robot, este procederá a girar la dirección a la que apunta 90º en sentido antihorario.
- Mensajes:
  - Objeto encontrado / Modificador encontrado: Mensaje de información avisando de que el robot procederá a girar en el sentido apropiado.
  - Objeto encontrado / Modificador no encontrado: Mensaje de error advirtiendo la falta de dirección a la que girar.
  - Objeto no encontrado: Mensaje de error avisando de la falta de objetos válidos.

B.2.14 Turn Off

- Acciones:
  - APAGA
  - APAGA_D
  - APAGA_I
  - SWITCH OFF
  - TURN OFF
- Objeto:
  - LAMPARA | LAMP
  - LAMPARA_DERECHA
  - LAMPARA_IZQUIERDA
  - TELEVISION
  - RADIO
- Modificadores:
  1. UNO | ONE | APAGA_D
  2. DOS | TWO | APAGA_I
  3. TRES | THREE
  4. ESTA | THIS
  5. ESA | AQUELLA | THAT
- Efectos en el espacio virtual:
  - LAMPARA | LAMP:

        - Si no se encuentra modificador y la luz del techo se encuentra encendida, se apaga.
        - Si se encuentra un modificador de tipo 1 y la lámpara uno o derecha se encuentra encendida, se apaga.
        - Si se encuentra un modificador de tipo 2 y la lámpara dos o izquierda se encuentra encendida, se apaga.
        - Si se encuentra un modificador de tipo 3, no produce cambios en el espacio virtual.
        - Si se encuentra un modificador de tipo 4, no produce cambios en el espacio virtual.
    - Objeto LAMPARA_DERECHA:
        - Si la lámpara uno o derecha se encuentra encendida, se apaga.
    - Objeto LAMPARA_IZQUIERDA:
        - Si la lámpara dos o izquierda se encuentra encendida, se apaga.
    - Objeto TELEVISION:
        - Si la televisión se encuentra encendida, se apaga.
    - Objeto RADIO:
        - Si la radio se encuentra encendida, se apaga.
    - No se encuentra objeto:
        - No produce cambios en el espacio visual.

- Mensajes:
    - LAMPARA | LAMP:
        - Modificador no encontrado / Luz del techo encendida: Mensaje indicando que se apagará la luz del techo.
        - Modificador no encontrado / Luz del techo apagada: Mensaje de error indicando que no se puede apagar la luz del techo.
        - Modificador tipo 1 encontrado / Lámpara uno encendida: Mensaje indicando que se apagará la lámpara uno.
        - Modificador tipo 1 encontrado / Lámpara uno apagada: Mensaje de error indicando que no se puede apagar la lámpara uno.
        - Modificador tipo 2 encontrado / Lámpara dos encendida: Mensaje indicando que se apagará la lámpara dos.
        - Modificador tipo 2 encontrado / Lámpara dos apagada: Mensaje de error indicando que no se puede apagar la lámpara dos.
        - Modificador tipo 3 encontrado / Luz del techo encendida: Mensaje indicando que se apagará la luz del techo.
        - Modificador tipo 3 encontrado / Luz del techo apagada: Mensaje de error indicando que no se puede apagar la luz del techo.
        - Modificador tipo 4 encontrado / Mensaje de error indicando que no se ha implementado aún esta función.
        - Modificador tipo 5 encontrado / Mensaje de error indicando que no se ha implementado aún esta función.
    - Objeto LAMPARA_DERECHA:
        - Lámpara uno encendida: Mensaje indicando que se apagará la lámpara uno.

    - Lámpara uno apagada: Mensaje de error indicando que no se puede apagar la lámpara uno.
  - Objeto LAMPARA_IZQUIERDA:
    - Lámpara dos encendida: Mensaje indicando que se apagará la lámpara dos.
    - Lámpara dos apagada: Mensaje de error indicando que no se puede apagar la lámpara dos.
  - Objeto TELEVISION:
    - Televisión encendida / Modificador no encontrado: Mensaje indicando que se apagará la televisión.
    - Televisión encendida / Modificador encontrado: Mensaje de advertencia indicando que se apagará la televisión y recordando que solo hay una televisión.
    - Televisión apagada: Mensaje de error indicando que no se puede apagar la televisión.
  - Objeto RADIO:
    - Radio encendida / Modificador no encontrado: Mensaje indicando que se apagará la radio.
    - Radio encendida / Modificador encontrado: Mensaje de advertencia indicando que se apagará la radio y recordando que solo hay una radio.
    - Radio apagada: Mensaje de error indicando que no se puede apagar la radio.
  - No se encuentra objeto:
    - Mensaje de error avisando de que no se ha encontrado un objeto válido.

B.2.15 Turn On

- Acciones:
  - ENCIENDE
  - ENCIENDE_D
  - ENCIENDE_I
  - TURN ON
  - SWITCH ON
- Objeto:
  - LAMPARA | LAMP
  - LAMPARA_DERECHA
  - LAMPARA_IZQUIERDA
  - TELEVISION
  - RADIO
- Modificadores:
  1. UNO | ONE | APAGA_D
  2. DOS | TWO | APAGA_I
  3. TRES | THREE
  4. ESTA | THIS

5. ESA | AQUELLA | THAT

- Efectos en el espacio virtual:
  - LAMPARA | LAMP:
    - Si no se encuentra modificador y la luz del techo se encuentra apagada, se enciende.
    - Si se encuentra un modificador de tipo 1 y la lámpara uno o derecha se encuentra apagada, se enciende.
    - Si se encuentra un modificador de tipo 2 y la lámpara dos o izquierda se encuentra apagada, se enciende.
    - Si se encuentra un modificador de tipo 3, no produce cambios en el espacio virtual.
    - Si se encuentra un modificador de tipo 4, no produce cambios en el espacio virtual.
  - Objeto LAMPARA_DERECHA:
    - Si la lámpara uno o derecha se encuentra apagada, se enciende.
  - Objeto LAMPARA_IZQUIERDA:
    - Si la lámpara dos o izquierda se encuentra apagada, se enciende.
  - Objeto TELEVISION:
    - Si la televisión se encuentra apagada, se enciende.
  - Objeto RADIO:
    - Si la radio se encuentra apagada, se enciende.
  - No se encuentra objeto:
    - No produce cambios en el espacio visual.

- Mensajes:
  - LAMPARA | LAMP:
    - Modificador no encontrado / Luz del techo apagada: Mensaje indicando que se encenderá la luz del techo.
    - Modificador no encontrado / Luz del techo encendida: Mensaje de error indicando que no se puede encender la luz del techo.
    - Modificador tipo 1 encontrado / Lámpara uno apagada: Mensaje indicando que se encenderá la lámpara uno.
    - Modificador tipo 1 encontrado / Lámpara uno encendida: Mensaje de error indicando que no se puede encender la lámpara uno.
    - Modificador tipo 2 encontrado / Lámpara dos apagada: Mensaje indicando que se encenderá la lámpara dos.
    - Modificador tipo 2 encontrado / Lámpara dos encendida: Mensaje de error indicando que no se puede encender la lámpara dos.
    - Modificador tipo 3 encontrado / Luz del techo apagada: Mensaje indicando que se encenderá la luz del techo.
    - Modificador tipo 3 encontrado / Luz del techo encendida: Mensaje de error indicando que no se puede encender la luz del techo.
    - Modificador tipo 4 encontrado / Mensaje de error indicando que no se ha implementado aún esta función.

    - Modificador tipo 5 encontrado / Mensaje de error indicando que no se ha implementado aún esta función.

- Objeto LAMPARA_DERECHA:
    - Lámpara uno apagada: Mensaje indicando que se encenderá la lámpara uno.
    - Lámpara uno encendida: Mensaje de error indicando que no se puede encender la lámpara uno.
- Objeto LAMPARA_IZQUIERDA:
    - Lámpara dos apagada: Mensaje indicando que se encenderá la lámpara dos.
    - Lámpara dos encendida: Mensaje de error indicando que no se puede encender la lámpara dos.
- Objeto TELEVISION:
    - Televisión apagada / Modificador no encontrado: Mensaje indicando que se encenderá la televisión.
    - Televisión apagada / Modificador encontrado: Mensaje de advertencia indicando que se apagará la televisión y recordando que solo hay una televisión.
    - Televisión encendida: Mensaje de error indicando que no se puede encender la televisión.
- Objeto RADIO:
    - Radio apagada / Modificador no encontrado: Mensaje indicando que se encenderá la radio.
    - Radio apagada / Modificador encontrado: Mensaje de advertencia indicando que se apagará la radio y recordando que solo hay una radio.
    - Televisión encendida: Mensaje de error indicando que no se puede encender la radio.
- No se encuentra objeto:
    - Mensaje de error avisando de que no se ha encontrado un objeto válido.